UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

PROYECTO FIN DE CARRERA

INTERFACES HOMBRE-MÁQUINA MEDIANTE TÉCNICAS DE SEGUIMIENTO DE OJOS Y RECONOCIMIENTO DE VOZ

Esther Guerrero Santana

JULIO 2014

INTERFACES HOMBRE-MÁQUINA MEDIANTE TÉCNICAS DE SEGUIMIENTO DE OJOS Y RECONOCIMIENTO DE VOZ

AUTOR: Esther Guerrero Santana

TUTOR: Pablo Varona Martínez

Grupo de Neurocomputación Biológica

Dpto. de Ingeniería Informática

Escuela Politécnica Superior

Universidad Autónoma de Madrid

Julio de 2014

Agradecimientos

En primer lugar, agradecer a mis padres, que me acompañaron y me dieron la oportunidad de poder estudiar y en segundo lugar dar las gracias a mi tutor de proyecto, Pablo Varona, por ayudarme a hacer este trabajo y por su dedicación.

Muchas son las personas que han contribuido a que por fin sea ingeniero, bueno, ahora graduada, y que, a pesar de los malos momentos, haya vivido los que probablemente hayan sido los mejores años de mi vida.

Cuando entré en esta carrera, no sabía qué era un puntero, ni siquiera sabía qué era programar. Menos mal que conocí a Jorge Santos, y me ayudó a comprender para qué servía un *printf* o cómo hacer *push* y *pop* en la pila, explicándomelo una vez tras otra desesperado.

Un año pasó, y ahí estaba Rafael Leira pasando sus tardes conmigo haciendo prácticas y ayudándome con la teoría de esas asignaturas que me quedaron en primero, intercambiando ideas por platos de macarrones y pollo.

También agradecer a Marcos Muñoz su apoyo incondicional, sus ánimos para terminar las cosas y sus noches sin dormir para poder lograrlo... Gracias por estar a mi lado durante este tiempo, ¡formamos un gran equipo!

A Verónica, Raquel y Cristina por estar siempre ahí, dispuestas a salir a darlo todo y a hacer planes y luego quedarnos en casa, y por todas nuestras risas, ideas disparatadas, fiestas, cotilleos varios y por supuesto por los viajes que hicimos y los que no hicimos…

A todos los amigos que he conocido a lo largo de la carrera, habéis sido mis compañeros de prácticas, clase, y comidas.

Y a mis chicas del cole que siempre hemos estado unidas aunque cada una haya estudiado una cosa distinta en cada punta de Madrid.

Por último, mencionar también el espíritu de colaboración de muchos profesores con los que he podido expresar opiniones e intercambiar experiencias, y a todos aquellos que me dieron clase, porque sin ellos no hubiera llegado hasta aquí: Pablo Varona, Carlos Aguirre, Julia Díaz, José Dorronsoro (enseñándonos si primero nos ponemos los calcetines o los pantalones para aprender algoritmos), Simone Santini, Iván Cantador, (que aunque es del atleti se le aprecia igual), Javier Aracil (con sus frases míticas: “Se masca la tragedia”), Manuel Sánchez Montañés, Rosa Carro, Álvaro Ortigosa, Idoia Alarcón (con sus cafés, tartas y galletas mañaneras, ayudándonos a no dormirnos primera hora de la mañana) y muchos otros más.

Palabras Clave

Seguimiento de ojos, caracterización de la actividad de un usuario, mirada, gesto ocular, interacción persona ordenador.

Resumen

En este proyecto se ha abordado el diseño y validación de un sistema (en este caso una red neuronal) para caracterizar la actividad que realiza una persona delante de un ordenador mediante el seguimiento ocular. El objetivo de este prototipo es diferenciar entre cuatro tipos de actividades: la lectura de un texto, el visionado de un vídeo tranquilo, donde aparece una playa durante 35 segundos, el visionado de un vídeo de acción también de 35 segundos y la navegación en la página web de la Universidad Autónoma de Madrid.

Para conseguir esta implementación, se utilizará la tecnología de *Eye Tracking* (seguimiento de los ojos) usando el dispositivo *Eye Tracker* de *Tobii X2-30*. Mediante un software de *Eye Tracking* se obtendrán las posiciones de la pantalla donde miran los ojos (coordenadas x e y) a lo largo del tiempo. Una vez obtenidas dichas coordenadas, se determinará la actividad realizada por el usuario a través del algoritmo de reconocimiento.

Lo primero que se llevará a cabo en este trabajo será acceder a los datos de seguimiento de la pupila del dispositivo de *Tobii*, para poder hacer uso de la tecnología *Eye Tracking*. Una vez entendido el funcionamiento y el manejo de dicha cámara, se pasará a utilizarla con distintos usuarios para generar ficheros con coordenadas de distintas acciones y así tener una pequeña base de datos que servirá como entrenamiento para la red neuronal.

Finalmente, se procederá a la validación del reconocimiento de las series temporales obtenidas en las distintas tareas de usuario registradas durante los experimentos.

Key words

Eye Tracking, Characterization of user activity, Gaze, Gaze gesture, Human computer interaction.

Abstract

This project deals with the design and validation of a system (in this case, a neural network) to characterize the activity performed by a person in front of a computer by means of eye tracking. The goal of this prototype is to discriminate between four types of activities: reading a text, viewing a 35-second peaceful video showing a beach, viewing a 35-second action video and surfing in the Universidad Autónoma de Madrid website.

To achieve this implementation, Eye Tracking technology will be used, as provided by the Tobii X2-30 Eye Tracker. By means of the Eye Tracking software, gaze screen positions (coordinates x and y) of the eye will be obtained over time. Once these coordinates are obtained, the user's activity will be determined through the recognition algorithm.

The first step that will take place in this work will be to access pupil tracking data from the Tobii device in order to apply the eye tracking technology. Once the operation of the camera is understood, it will be applied to several users to create files containing coordinates corresponding to the time series from different user tasks and thus have a small database available that will work as the training set for the neural network.

Finally, we will proceed to validate the recognition of the time series obtained during the experiments for the different user tasks.

GLOSARIO

HCI: Human Computer Interaction.

HTM: Hierarchical Temporal Memory.

IR: Infrared.

RNA: Red Neuronal Artificial.

EYE TRACKING: Tecnológica que evalúa el punto donde se fija la mirada

GAZE GESTURES: Gesto con la mirada.

ECM: Error Cuadrático Medio.

CLUSTERING: Proceso de agrupar objetos similares: *clusters*

CLUSTER: Colección de objetos.

ÍNDICE DE CONTENIDOS

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

ÍNDICE DE ECUACIONES

1 INTRODUCCIÓN

1.1 Motivación

La posibilidad de interactuar con una máquina sin necesidad de hacer uso de las interfaces típicas destinadas a tal efecto como son el teclado y el ratón, ofrece una atractiva alternativa para desarrollar y llevar a cabo estudios orientados a una mejor calidad de vida para aquellas personas que no pueden hacer uso de sus manos por discapacidad física, amputaciones de extremidades, lesiones crónicas o enfermedades como la esclerosis múltiple, etc. Por otro lado, algunos tipos de señalización biológica también pueden utilizarse en aplicaciones de biometría y de caracterización de la actividad que realiza una persona delante del ordenador.

En este contexto, determinar qué hace un sujeto frente a un ordenador a partir del movimiento de sus ojos puede tener múltiples aplicaciones. Por un lado da a conocer qué es lo que se está mirando y por ello, qué cosas son las que nos interesan cuando se mira, por ejemplo, en una página web o cuando se está realizando una navegación o búsqueda. Por otro lado, el patrón de mirada durante la realización de una actividad puede ser específico de esa actividad.

1.2 Objetivos

El objetivo principal del proyecto es detectar qué está haciendo un usuario frente a un ordenador dentro de un conjunto estereotipado de tareas. Gracias a los gestos y movimientos de los ojos que los sujetos realizan, se puede abordar la determinación de qué acción están realizando entre cuatro posibles opciones: leer un texto, ver un video tipo 1, ver un video tipo 2 o navegar por internet.

Para realizar esta tarea, se dispondrá de la cámara *Tobii X2-30*, que es la que se encarga de captar la pupila del ojo y ofrecer las coordenadas de éste.

En la implementación, se desarrollará en código C un programa para generar las series temporales a partir de los datos que produce la cámara utilizada, además de scripts para Matlab para facilitar el uso de la red neuronal para clasificar la actividad del usuario.

En la fase de pruebas, se validará el funcionamiento de la red neuronal realizando pruebas con usuarios. En este caso se tomarán 18 sujetos, los cuales han realizado cuatro acciones distintas. Dichos usuarios formarán parte del entrenamiento, mientras que para la parte de test, se elegirán otros 5 voluntarios para que realicen cualquiera de las cuatro acciones y proceder así a clasificar los datos. En total, se utilizan 72 series temporales para el entrenamiento y 20 para el test de validación.

1.3 Organización de la memoria

La memoria consta de los siguientes capítulos:

- **Capítulo 1**. Introducción: Motivación, objetivos y organización de la memoria.

- **Capítulo 2**. Estado del arte: Repaso de las interfaces tradicionales, recorrido a lo largo de la historia de las interfaces basadas en *Eye Tracking* y presentación de los *Gaze Gestures*.

- **Capítulo 3**. Diseño: Construcción del soporte y descripción del proceso de reconocimiento de gestos.

- **Capítulo 4**. Desarrollo: Diferentes formas de abordar la clasificación, explicaciones de las series temporales y su clasificación

- **Capítulo 5**. Resultados: Descripción de las pruebas y resultados obtenidos a partir de las mismas.

- **Capítulo 6**. Conclusiones.

- **Capítulo 7**. Trabajo futuro.

2 EL ESTADO DEL ARTE

2.1 Introducción a HCI

La interacción hombre-máquina (Human Computer Interaction, HCI), se define como la disciplina que estudia el intercambio de información entre las personas y las computadoras, es decir, la forma que tiene una persona de interactuar con un ordenador.

Desde el origen de las computadoras, la forma más habitual para comunicarse con un ordenador ha requerido el uso de las manos, debido a que el ser humano, hace uso de estas extremidades desde el inicio de los tiempos para cualquier uso cotidiano; desde trabajar a escribir o comer. Debido a esto, las interfaces como el teclado o el ratón han sido y son prácticamente universales para la comunicación con cualquier tipo de máquina.

Es por ello que la tecnología táctil ha tenido mucho auge y ha cobrado un nuevo sentido gracias al aumento de dispositivos como *Smartphones* o tabletas, que se basan en el uso de los gestos dactilares para todo tipo de funciones, como bloquear o desbloquear el teléfono móvil mediante un patrón de desbloqueo. Este simple gesto, es algo que un porcentaje elevado de personas puede hacerlo, pero ¿qué ocurre con el porcentaje minoritario? ¿puede una persona incapacitada hacer uso o interactuar con una máquina u ordenador? Enseguida nos damos cuenta que no todo el mundo puede utilizar sus manos y hacer uso de un ordenador, es por ello que se realizan distintos estudios para poder proporcionar otros mecanismo de interacción con el ordenador y facilitar la vida de las personas discapacitadas.

Una posible solución a este problema, es el **control por voz**, como es por ejemplo los que se utilizan para controlar un sistema de navegación basado en el GPS o la acción de llamada de un *smartphone*.

Otra opción muy interesante y complementaria es el **control mediante el seguimiento de ojos**: *Eye Tracking*. Se trata de una tecnología que posibilita la interactuación con los ojos, consiguiendo por ejemplo, que un cursor apunte al lugar del monitor del ordenador al cual se esté mirando. Las técnicas de *eye-tracking* se aplican a una amplia variedad de disciplinas y áreas de estudio, como puede ser el márketing, la publicidad o la investigación médica (Olsen & Marshall, 2011; D Rozado, Moreno, San Agustin, Rodriguez, & Varona, 2014). En este trabajo se propone el uso de estas técnicas para la caracterización de la actividad que realiza una persona delante del ordenador.

2.2 Interacción Hombre-Máquina

A lo largo de la historia de la informática, los humanos hemos tenido que adaptarnos al uso de distintos tipos de dispositivos, los cuales a su vez han sido adaptados a nuestras necesidades.

Desde 1900 hasta los años 1950 y 1970, los primeros medios para introducir información e instrucciones a un ordenador fueron las **tarjetas perforadas**. Éstas datan del 1725 y en su versión más habitual consistían de una lámina de cartulina con información digital en forma de perforaciones, es decir, representación binaria (presencia o ausencia de agujeros) en posiciones determinadas. (Guerrero, 2013)

Con el avance y el paso de los años, las interfaces hombre-máquina han evolucionado. Después de las tarjetas perforadas, IBM creó el primer **ordenador**, progenitor de la plataforma hardware: **teclado**, **pantalla** y **ratón**.

El teclado tiene su origen en las máquinas de escribir y aparece alrededor de 1980 como interfaz para el ordenador. Es quizás el componente menos valorado, pero el que más echamos en falta cuando se estropea. Hoy en día, los avances de las tecnologías han hecho posible que el teclado sea una pieza integrada en distintos tipos

de superficie. En concreto, ha pasado de una telefonía móvil con teclas a una telefonía con un teclado virtual integrado.

El ratón, diseñado en la década de los 60, facilita el manejo del entorno gráfico del ordenador, ya que se trata de un dispositivo apuntador para poder movernos por la pantalla. En los dispositivos móviles de hace unos años, esto se hacía con el teclado. Ahora los gestos táctiles son los que manejan el entorno a través de la pantalla.

Si juntamos todas estas peculiaridades, obtenemos lo que a día de hoy se llama la **interfaz táctil**, como las utilizadas en los *smatphones* o *tablets.* Se trata de pantallas táctiles que no necesitan de un hardware como el ratón y el teclado, pues con el manejo de un dedo puede llegar a controlarse.

Con la llegada de estos teléfonos inteligentes, tabletas o GPS, la **voz** como interfaz hombre-máquina, empieza a tener mayor importancia, debido a la eficacia de las aplicaciones de reconocimiento de voz que facilitan estas labores. Solicitar indicaciones para ir a algún lugar, es ahora más sencillo simplemente con decir la dirección en lugar de escribirla.

Otra de las posibilidades, es poder interactuar con la máquina a través de los **ojos**. Existen diferentes métodos y dispositivos para llevar esto a cabo:

- Head-mounted o montaje en la cabeza: Este método lleva su montaje en la cabeza para evitar las posibles invariantes. En la Ilustración 1 se puede ver un ejemplo de este dispositivo.

- *Tobii X2-30*: Este método de disposición fija tiene en cuenta las invariantes, como por ejemplo el movimiento de la cabeza, característica que el dispositivo anterior mencionado no tiene y para evitar este tipo de invariantes se coloca en la cabeza.

- *Eye Tribe: The Eye Tribe* es una *startup* danesa que desarrolla tecnología de seguimiento de ojos. Permite controlar algunos aspectos del dispositivo con el

movimiento de los ojos, como por ejemplo jugar a *Fruit Ninja.* Este juego consiste en una tarea sencilla: partir en dos frutas que van saliendo por la pantalla. (Tribe, 2014)

De este modo, los sistemas se pueden clasificar como: a distancia o montado en la cabeza. En sistemas remotos, la cámara y las fuentes de luz se colocan a una distancia del usuario, normalmente alrededor la pantalla del ordenador, mientras que en los sistemas montados en la cabeza los componentes se colocan en la cabeza del usuario, por lo general montado en un casco o unas gafas.

Este tipo de interacciones con el ordenador supone un salto cuantitativo respecto al periodo no muy lejano cuando las órdenes a los ordenadores se introducían mediante líneas de comandos en una pantalla negra. A partir de aproximadamente 1984, los ordenadores se llenaron de colores, iconos, ventanas, menús… gracias a la interfaz gráfica. Aunque esto hizo de las computadoras algo más amigables, la relación sigue limitada a una interacción hombre-máquina basaba en la comunicación mediante texto, voz, seguimiento de ojos.

2.3 Tecnologías a utilizar: Eyes Tracking y Gaze gestures

¿Qué es ***Eye Tracking***? *Eyes Tracking* es una solución tecnológica que evalúa el punto donde se fija la mirada, es decir, permite realizar un seguimiento de ojos, extrayendo información del usuario analizando sus movimientos oculares. (“Eyetracking,” 2013)

¿Qué es ***Eye Tracker***? Un *Eye Tracker* es la herramienta con la que se realiza la acción explicada anteriormente para determinar la posición a la que miran los ojos en una pantalla (A. Duchowski, 2007). Se trata de un dispositivo con una luz infrarroja y una cámara que graba cómo ese haz de luz refleja en el ojo del usuario. La luz infrarroja mejora el contraste de la imagen y produce una reflexión sobre la córnea, conocido como reflejo corneal o brillo. Esta herramienta, permite registrar el

movimiento ocular con bastante exactitud y determinar qué recorrido ha realizado la persona con los ojos.

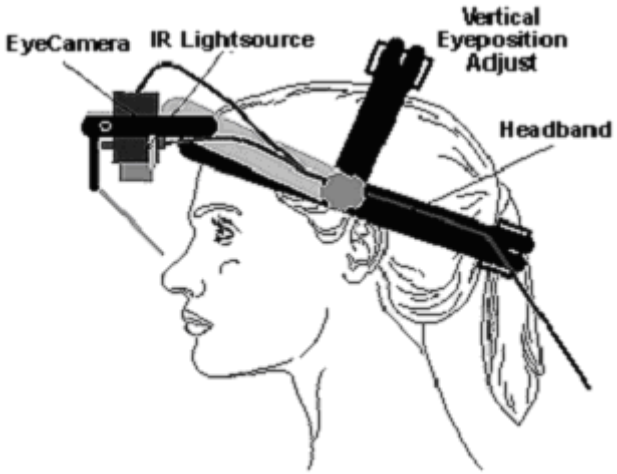


Ilustración 1. Montaje de un *eye tracker* sobre la cabeza.

¿Qué información se extrae cuando se realiza *Eye Tracking*? La información que se puede obtener va desde el punto donde se está mirando, qué le llama la atención al usuario, saber dónde poner el contenido de valor para el usuario, incluso sus intenciones y estados de ánimo. ("¿Qué es el 'Eye Tracking' y para qué nos sirve?," 2012)

Por otro lado, el concepto de *gaze tracker* se refiere a utilizar los datos del seguimiento ocular para determinar gestos oculares que puedan utilizarse como comandos al ordenador (David Rozado, Agustin, Rodriguez, & Varona, 2012).

Se ha probado dos tipos de E*ye Tracker:* uno de tipo head-mounted como muestra la Ilustración 1 y otro de disposición fija, que es el usado en este proyecto. Para el primer tipo (head mounted), haciendo uso de ambas herramientas, (*Eye tracker* y *Gaze Tracker*), se ha utilizado una gorra a la que se le ha enganchado una cámara. Ésta cámara enfoca al ojo, captando así la pupila. Las luces infrarrojas que se han usado, no estaban incorporadas en la cámara, sino que se han colocado encima de una mesa alrededor de un ordenador, apuntado a los ojos. La imagen de la cámara del *Eye Tracker*, se refleja en el programa o herramienta *Gaze Tracker*. De este modo, se ha

movido el cursor a los diferentes puntos de la pantalla donde se está mirando, sin necesidad de hacer uso del ratón y las manos.

¿Dónde se aplica la tecnología *Eye Tracking*? Esta tecnología tiene diferentes aplicaciones (A. Duchowski, 2007; A. T. Duchowski, 2002; Granka, Joachims, & Gay, 2004; Lee, Lim, Hwang, & Im, 2013; Olsen & Marshall, 2011; Poole & Ball, 2005), ya que es de fácil manejo, con una interacción rápida ("Diferentes aplicaciones de la tecnología Eye Tracking," 2004):

- Área de diseño y publicidad: Uno de los entornos con mayor facilidad de uso, es para hacer estudios de mercado y publicidad, analizando por ejemplo las partes de un anuncio que llaman más la atención a los consumidores.

- Usabilidad de páginas web: Saber qué es lo que mira un usuario y qué es lo que más le interesa.

- Áreas clínicas: Otro campo muy interesante son áreas clínicas, como la accesibilidad para personas con movilidad reducida, las cuales sólo pueden interaccionar de modo visual o las operaciones mediante láser para eliminar la miopía, la hipermetropía o el astigmatismo.

- Mejora del rendimiento deportivo: Con objetivo de marcar canasta o un gol, existen entrenamientos ayudando a los jugadores a que aprendan a fijarse en determinadas zonas para así mejorar el tiro.

- Telefonía móvil: Hoy en día, los *smartphones* ya llevan incorporados un sistema *Eyes Tracking*, básicos como el Samsung Galaxy S3. Gracias a esta tecnología, se evita el bloqueo del teléfono móvil si detecta que el usuario está mirando la pantalla.

¿Qué es un ***Gaze Gestures***? *Gaze Gesture* es un gesto con la mirada. Se trata de una secuencia de trazos, donde, diferentes patrones de estos trazos definen diferentes

gestos. Lo que se hará en este proyecto, será realizar diferentes actividades definiendo los movimientos oculares realizados dentro de un período de tiempo limitado, en un rango y área particulares, lo que se utilizará para indicar una intención particular del usuario (David Rozado, Agustin, et al., 2012; David Rozado, Rodriguez, & Varona, 2012a).

2.4 Métodos para detectar secuencias en gestos oculares

Existen varios métodos para detectar eventos en el reconocimiento de ojos como algoritmos de clasificación, redes neuronales, algoritmos para reconocer secuencias (HTM), etc. A continuación se explican algunos de ellos.

2.4.1 Alineamiento de secuencias (Algoritmo de *Needleman-Wunsch*)

El alineamiento de secuencias es un método utilizado para identificar la similitud entre dos o más secuencias. Se utiliza comúnmente en análisis de secuencias presentes en lenguaje natural, datos financieros y biológicos tales como secuencias de ADN o ARN. En la alineación de secuencias, el grado de similitud entre dos secuencias se puede deducir de la cantidad de elementos similares que ocupan la misma posición en una secuencia, el número de elementos diferentes y el número de espacios necesarios para que las secuencias se alineen.

Para el reconocimiento de gestos con la mirada realizados por una secuencia de movimientos sobre diferentes áreas de la pantalla, se utilizó una medida de similitud entre las secuencias. La similitud de secuencia se calcula utilizando el algoritmo de *Needleman-Wunsch* que emplea programación dinámica para calcular una medida de similitud entre dos secuencias, donde una SeqA es la secuencia del gesto realizado y SeqB la secuencia del gesto que se ha de reconocer. (David Rozado, Agustin, et al., 2012; David Rozado, Rodriguez, & Varona, 2012b).

2.4.2 Memoria temporal Jerárquica (HTM)

HTM es un paradigma de reconocimiento de patrones bioinspirado que ha sido utilizado con éxito en varias aplicaciones de aprendizaje automático (George & Hawkins, 2009). HTM pretende capturar las propiedades estructurales del neocórtex. El neocórtex o neocorteza de los mamíferos es la base del pensamiento inteligente en el cerebro, tales como la visión, tocar o escuchar.

Programar HTM implica entrenar mediante la exposición a un flujo de datos. Cuantos más datos se tengan, mejores resultado se obtendrán. Dado que HTM modela los detalles arquitecturales del neocórtex, puede considerarse como una red neuronal. Las redes son entrenadas con muchos tipos de datos variados y se confían en el almacenamiento de grandes grupos de patrones y secuencias. Así pues, HTM es un sistema basado en la memoria.

Una red HTM se organiza de forma jerárquica, lo que aporta una gran eficiencia. Está compuesta por regiones (unidad principal de memoria y predicción), las cuales representan un nivel. La organización jerárquica reduce el tiempo de entrenamiento y el uso de memoria, ya que los patrones aprendidos en cada nivel son reutilizados en los niveles superiores. Por ejemplo, cuando se aprende una palabra nueva, no hace falta reaprender las letras o sílabas.

Por tanto, el trabajo de los algoritmos HTM es aprender las secuencias temporales de un conjunto de datos de entrada, hacer inferencia (evaluar) y predecir.

2.4.3 Redes Neuronales

Una RNA (Red Neuronal Artificial) es un esquema de computación distribuida inspirada en la estructura del sistema nervioso de los seres humanos. Está formada por neuronas conectadas entre sí, procesando y propagando su información a partir de su

entrada para producir una salida. En este caso, las neuronas trabajarán para producir como salida la clasificación esperada.

¿Qué es una Red Neuronal FeedForward Backprograpation?

Una red neuronal *feedforward*, tiene un aprendizaje supervisado, es decir, se tienen datos de entrada y datos de salida. Por tanto, los parámetros de la red conocidos son estimados a partir de un conjunto de patrones de entrenamiento compuesto por patrones de entrada y salida. Así pues, el ajuste de la red se produce como resultado de la estimación de los parámetros.

El término *backpropagation* se refiere, a que durante el aprendizaje (de tipo supervisado) hay un ajuste de pesos, los cuales se van comparando con la salida esperada para minimizar el error (Ecuación 2). Este error se propaga hacia atrás y se vuelven a calcular los pesos, hasta que entre una etapa y otra, no haya habido ningún cambio de los mismos.

En este trabajo se trabajará con este tipo de red neuronal para detectar las actividades que llevan a cabo los usuarios.

3 DISEÑO

A continuación se explica el diseño de este proyecto: la construcción, montaje y estructura del *eye tracker*, así como los materiales necesarios a utilizar, y cómo llevar a cabo la comunicación entre el *eye tracker,* el ordenador y el usuario.

3.1 Construcción, montaje y estructura

Lo primero que se pensó para llevar a cabo el experimento, fue construir nuestro propio dispositivo Eye Tracker. Consistía en una cámara como la que se ve en la Ilustración 3, cuya base tiene forma de pinza. Esta cámara es enganchada hacia abajo a la visera de una gorra, enfocando de este modo al ojo. Además, fue necesario el uso de dos focos de luces IR como los de la Ilustración 2 para evitar el reflejo de otras luces.

Ilustración 3. Cámara Sandberg NightCam 2

Ilustración 2. Foco de luz IR

Debido a los problemas con la herramienta necesaria para capturar las coordenadas del ojo y realizar un correcto calibrado, se pasó a una segunda opción, utilizando otro tipo de cámara de disposición fija: *Tobii X2-30*.

3.1.1 Cámara

La cámara utilizada finalmente en este trabajo, es la que se muestra en la Ilustración 4. *Tobii* es el líder en fabricación y desarrollo de tecnologías de seguimiento ocular y el control de los ojos. Se trata de una tecnología que hace posible que los equipos sean controlados por los usuarios simplemente con la mirada, es decir, los usuarios pueden controlar el ordenador con los ojos, sin importar el color, si usan lentes de contacto, sin importar los movimientos de la cabeza o condiciones de iluminación. *Tobii* convierte el movimiento del ojo en un cursor, suplantando así, el ratón del ordenador. (Tobii, 2012)

Ilustración 4. Cámara Tobii X2-30

Ésta cámara lleva incorporadas las luces infrarrojas y tiene una gran precisión. Basta con colocarla debajo de la pantalla del ordenador, con algún soporte si fuera necesario, de tal modo que enfoque lo mejor posible a los ojos. *Tobii X2-30 Eye Tracker* tiene más tolerancia a los movimientos de la cabeza que otros rastreadores oculares. Los participantes pueden moverse libremente de forma natural. Es un sistema con cámara dual con selección automática de brillo ofreciendo un seguimiento de los dos ojos (Technology, 2013).

Lo primero que hay que hacer es instalar el driver. Habrá que seleccionar el tipo *eye tracker* (conectándolo con el puerto USB) y seleccionar el tipo de pantalla con el que se esté trabajando. Por último, habrá que hacer una correcta colocación de la cámara con la pantalla ajustándola respecto a unas líneas (mapa para visualizar el monitor, ver Ilustración 5). Este paso es muy importante, ya que si la cámara se mueve o descoloca, hay que volver a realizar estos pasos.

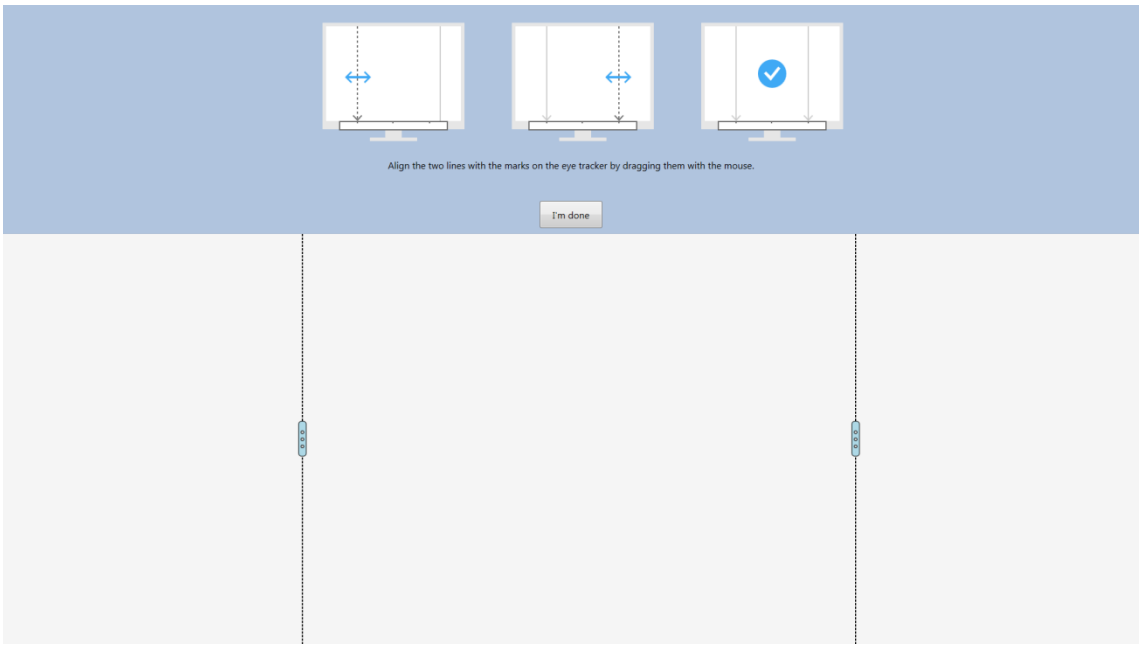


Ilustración 5. Mapa de visualización

Una vez instalado, el seguimiento es completamente automático con un procedimiento de calibración rápido. De este modo, cuando ejecutamos el *Eyetrackerbrowser.py* en la consola se muestra una pantalla donde se podrá ver si la cámara está bien colocada o no, dependiendo de si los ojos se ven reflejados en el programa como muestran las tres imágenes siguientes (Ilustración 6, Ilustración 7 e Ilustración 8).

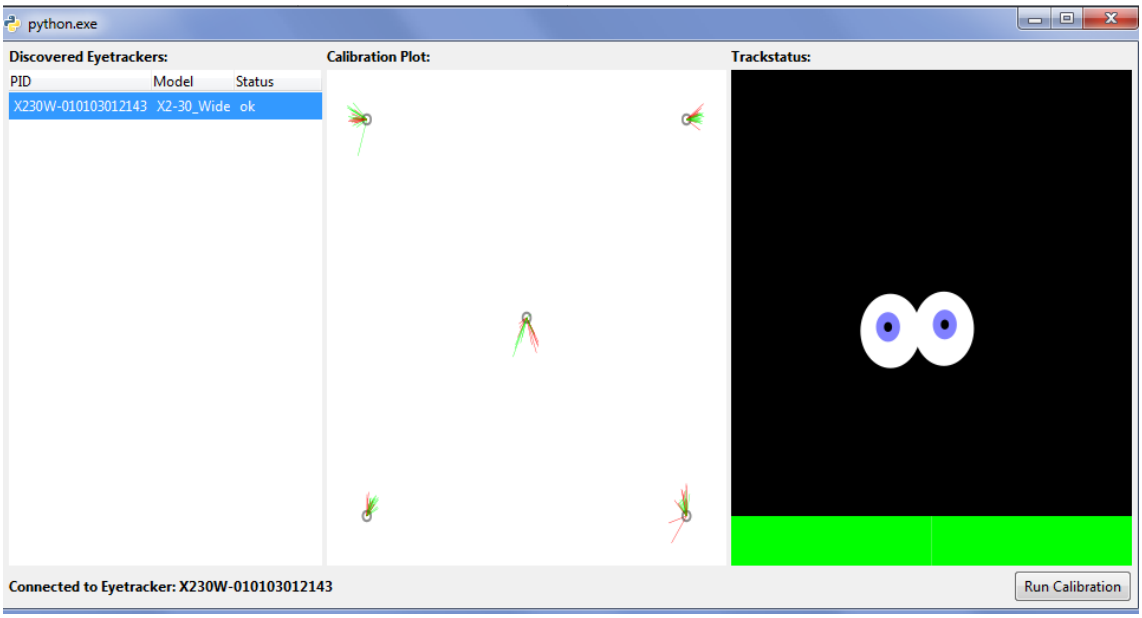


Ilustración 6. El usuario mira al frente

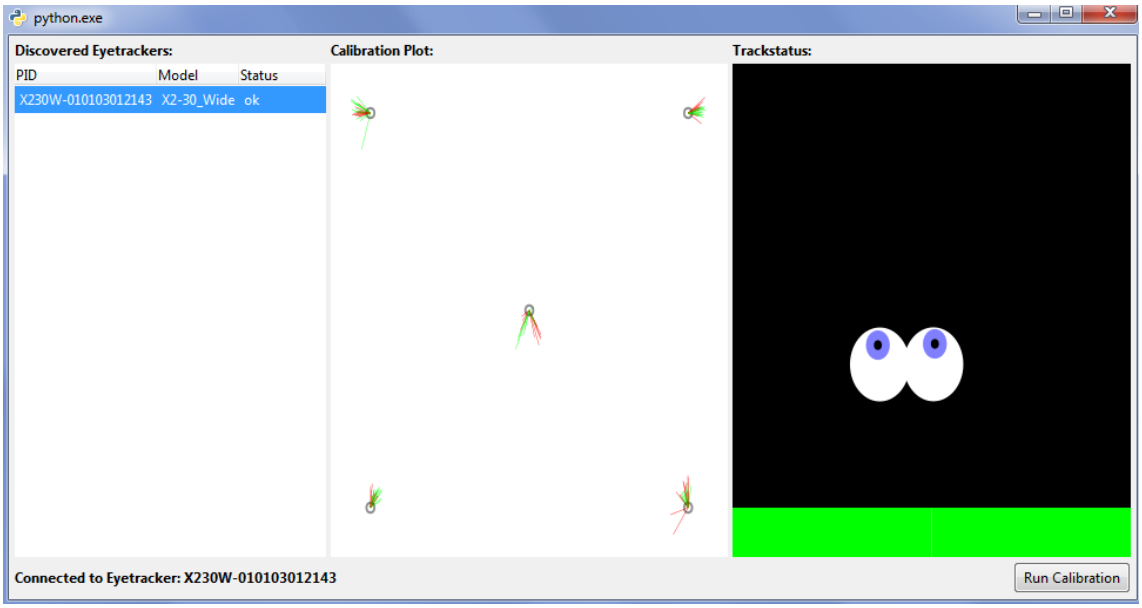


Ilustración 7. El usuario mira hacia arriba

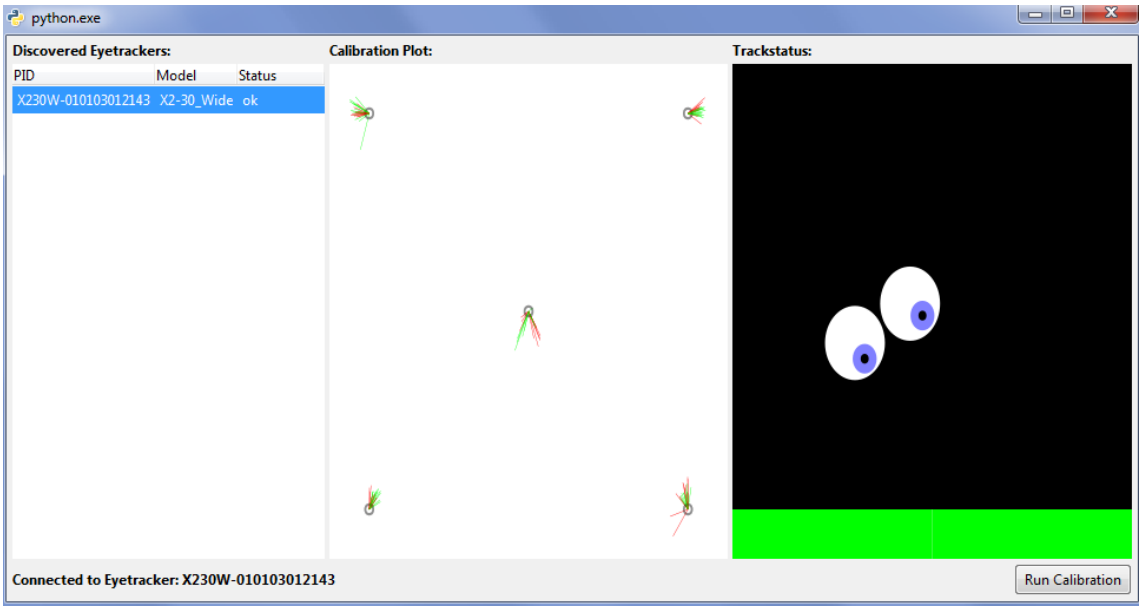


Ilustración 8. El usuario tiene la cabeza girada hacia la izquierda y la mirada abajo

Después de ver que la cámara enfoca correctamente a los ojos, se podrá calibrar, dándole a "*Run Calibration*". Aparecerá una pantalla en blanco en la que irán apareciendo una secuencia de círculos a los que habrá que ir mirando y seguir con los ojos. Tras este proceso, volveremos a la pantalla anterior, donde se verá el resultado de la calibración, como muestran la Ilustración 6, Ilustración 7 e Ilustración 8, en la zona central. Si se considera necesario, se puede volver a realizar una nueva calibración.

Una vez conseguida una buena calibración, se ejecutará el programa (*main.py*), el cual irá generando coordenadas (X e Y), a medida que el ojo realiza distintos movimientos y mira a distintas partes de la pantalla. De este modo, se sabrá qué puntos de la pantalla han sido los que el usuario ha mirado.

3.1.2 Luces Infrarrojas

La gran mayoría de la tecnología *Eye Tracking* necesita una iluminación infrarroja. Estas luces infrarrojas (IR), emiten una radiación electromagnética y térmica de mayor longitud de onda que la luz visible, proporcionando así una luz más estable. ¿Por qué usar luces IR? Esta luz ayuda a eliminar los reflejos producidos por otras fuentes de luz externas sin causar distracción al usuario ya que es invisible para el ojo humano.

Para nuestro dispositivo definitivo, no ha sido necesario usar ningún foco de luz IR, ya que la cámara *Tobii X2-30* utilizada, incorpora este tipo de tecnología.

3.2 Proceso de reconocimiento de gestos

Para reconocer un gesto, primero el usuario deberá realizar una de las cuatro actividades propuestas (ejecución del gesto), por ejemplo leer. Este gesto tiene unas características peculiares, ya que se trata de un movimiento que va de izquierda a derecha y de arriba a abajo. La cámara, en este caso *Tobii*, capta el movimiento que realiza el usuario con los ojos (imágenes del ojo). Es en este momento donde se hace uso de la tecnología "*eye tracking*", es decir, se evalúa el punto (compuesto de dos coordenadas: x e y) donde se fija la mirada. El gesto, compuesto por una serie de coordenadas es procesado y clasificado. La clasificación hace que sepamos a qué grupo pertenece el gesto realizado, (en este caso leer) y poder identificarlo.

Ilustración 9. Pasos del reconocimiento de gestos

3.3 Matlab

Matlab es un lenguaje de alto rendimiento para la informática técnica. Se integra computación, visualización y programación en un entorno fácil de usar.

Se trata de un sistema interactivo cuyo elemento básico para tratar los datos es una matriz. Esto permite resolver muchos problemas de computación técnica, en una fracción del tiempo que se tardaría en escribir un programa en un lenguaje no interactivo escalar como C.

Matlab ofrece una familia de soluciones de aplicaciones específicas denominadas cajas de herramientas (colecciones completas de funciones) que permiten aprender y aplicar la tecnología especializada. Éstas cajas incluyen el procesamiento de señales, sistemas de control, las redes neuronales y muchos otros. En este caso, se usará Matlab para probar una red neuronal, ya que permitirá trabajar más rápidamente que con C y más cómodamente.

4 DESARROLLO

En primer lugar, se han reclutado un conjunto de voluntarios para que realicen cuatro tipos de pruebas distintas. Cada una de estas pruebas corresponderá a distintas actividades oculares a clasificar, lo que se llamarán "clases". Con cada sujeto se calibrará la cámara y se realizarán las pruebas mientras que el sistema irá extrayendo información de la actividad del usuario analizando sus movimientos oculares. Esos movimientos quedarán guardados en un fichero en forma de series temporales como coordenadas X e Y. Cada fichero representa a una clase (uno por cada prueba). Las clases son representadas en binario, habiendo dos opciones para clasificar, las cuales se explican a continuación.

Para llevar a cabo la clasificación de los datos, se ha utilizado una red neuronal (*feedfoward backpropagation*).

4.1 Codificación de las clases: Dos neuronas en la capa de salida

La primera opción probada y elegida para codificar la clasificación es la que resulta con dos neuronas en la capa de salida de la red. Esto es, se tienen cuatro opciones o clases para la clasificación: leer, ver video1, ver video2 y navegar. ¿Cuántas neuronas se necesitan para poder clasificar? La respuesta es sencilla. Se necesita un número que, elevándolo al 2, devuelva 4, es decir $2^x = 4$. La red neuronal tendrá por tanto, dos neuronas en la capa de salida para poder clasificar cuatro clases.

Clase	Actividad
00	Leer texto
01	ver video tipo 1
10	ver video tipo 2
11	navegar por internet

Tabla 1. Clasificación binaria de las actividades

La opción final de la clasificación de las distintas acciones o actividades es la que se muestra en la Tabla 1.

Debido a que se ha elegido una clasificación binaria de ceros y unos, la función de transferencia es una sigmoide binaria con el siguiente aspecto y ecuación:

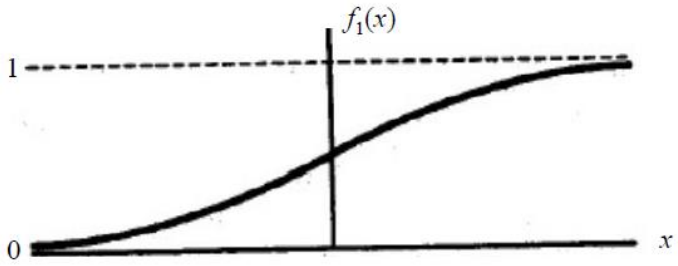


Ilustración 10. Sigmoide binaria

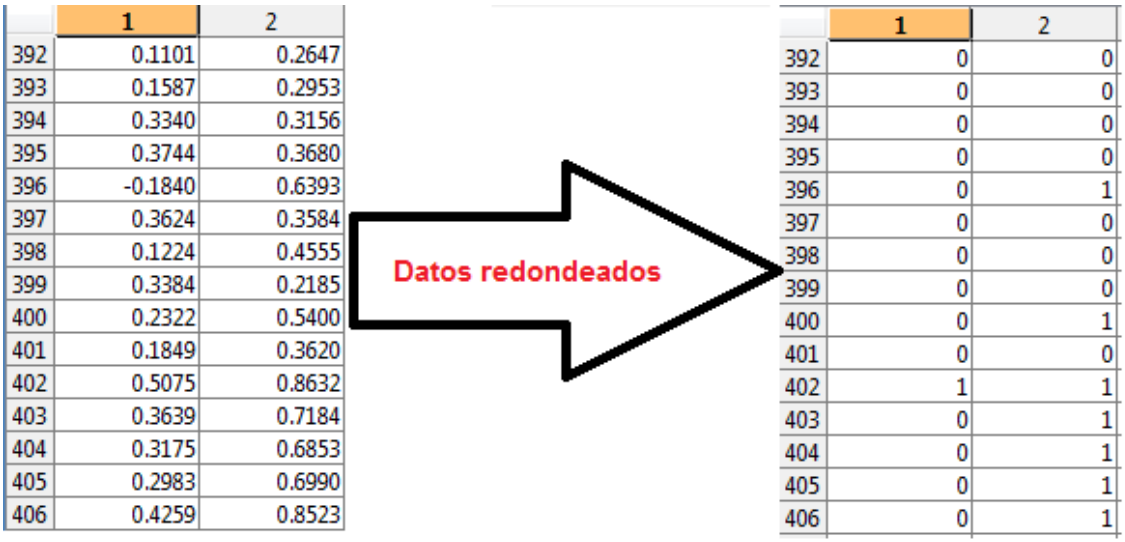
$$f_1(x) = \frac{1}{1 + \exp(-x)}$$

$$f_1'(x) = f_1(x)[1 - f_1(x)]$$

Ecuación 1. Función Sigmoide binaria

La función de transferencia es un modelo matemático (Ecuación 1) que relaciona la respuesta de la red neuronal a la entrada, es decir, la función genera la señal de salida de la neurona a partir de señales de entrada.

Por tanto, como se puede ver en la Ilustración 10, la función de transferencia toma valores cercanos a 0 y a 1, es por eso que la red no devolverá ni 0 ni 1. La red neuronal clasificará los datos según lo aprendido, haciendo que cada salida (cada fila) tome valores cercanos a 0 o 1. Para obtener la salida correcta, simplemente basta con hacer una aproximación y redondear al número más cercano (ver Ilustración 11).



	1	2
392	0.1101	0.2647
393	0.1587	0.2953
394	0.3340	0.3156
395	0.3744	0.3680
396	-0.1840	0.6393
397	0.3624	0.3584
398	0.1224	0.4555
399	0.3384	0.2185
400	0.2322	0.5400
401	0.1849	0.3620
402	0.5075	0.8632
403	0.3639	0.7184
404	0.3175	0.6853
405	0.2983	0.6990
406	0.4259	0.8523

	1	2
392	0	0
393	0	0
394	0	0
395	0	0
396	0	1
397	0	0
398	0	0
399	0	0
400	0	1
401	0	0
402	1	1
403	0	1
404	0	1
405	0	1
406	0	1

Ilustración 11. Datos redondeados

Una vez se tiene la salida con ese formato, bastará con contar el número de pares que aparece más veces (0 0, 0 1, 1 0 y 1 1) para saber qué clase es la que ha aparecido más veces en la salida, conociendo cual es la ganadora, y sabiendo así la actividad que ha realizado el usuario con el movimiento de sus ojos.

Por ejemplo, si los datos con los que se trabajasen fueran únicamente los datos que muestra la Ilustración 11, la red neuronal devolvería 0 0, sabiendo que los datos que contiene el fichero pasado a la red una vez entrenada, son de un usuario que ha estado leyendo. Esto es así, porque la clase 0 0 aparece ocho veces, mientras que la clase 0 1 se muestra seis veces, la 1 1 una única vez y la clase 1 0 ni siquiera aparece.

4.1.1 Alternativa: Cuatro neuronas en la capa de salida

Otra alternativa podría tener 4 neuronas en la capa de salida. Con esta codificación, cuando se realiza una de las cuatro acciones o pruebas posibles, se activará un 1 en la casilla que corresponda a cada neurona de salida, sabiendo así que la acción realizada es la que marca la casilla con el 1. La Tabla 3 muestra una visión más clara de lo explicado.

Clase	Actividad
1000	Leer texto
0100	ver video tipo 1
0010	ver video tipo 2
0001	navegar por internet

Tabla 2. Segunda clasificación binaria de las actividades

Pruebas	Clase 1: Leer	Clase 2 : Ver video tipo 1	Clase 3: Ver video tipo 2	Clase 4: Navegar
Prueba 1	**1**	0	0	0
Prueba 2	0	**1**	0	0
Prueba 3	0	0	**1**	0
Prueba 4	0	0	0	**1**

Tabla 3. Clasificación de las distintas pruebas

Para este caso, se necesitarán más conexiones en la red y crear el fichero de series temporales. Este fichero tendrá el doble de atributos, ya que se tiene el doble de neuronas en la capa de salida.

La red neuronal clasificará los datos según lo aprendido, es decir, cada fila tendrá valores cercaos a 0 o 1. En este caso no basta con hacer una aproximación ya que ahora se quiere que únicamente haya un solo 1 en cada una de las filas, es decir: 1000, 0100, 0010 o 0001, y no algo como: 1100 o 0101. Para que esto no suceda habrá que coger el máximo de cada fila, es decir, el que más cerca esté a 1 y ver en qué columna está (columna 1, 2, 3 o 4). De esta forma la columna que contenga el valor más cercano a 1 se aproximará a 1, y el resto estarán a 0, sabiendo así la clase a la que pertenecen los datos (Ilustración 12).

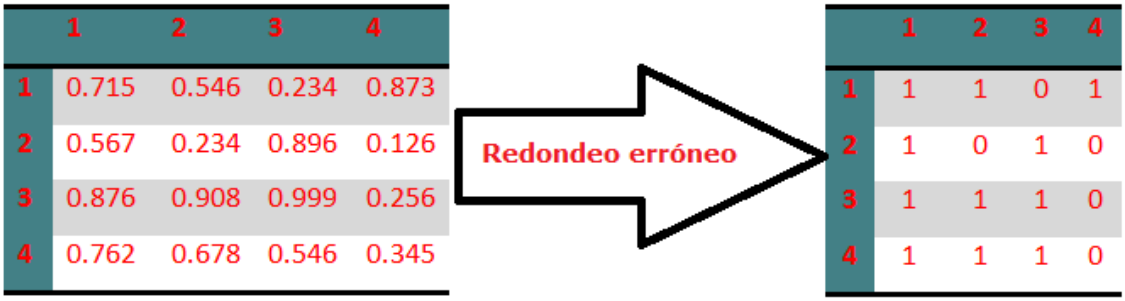


Ilustración 12. Redondeo erróneo

La correcta aproximación sería como muestra la Ilustración 13. Con este pequeño ejemplo y suponiendo que se tuvieran solo estos pocos de datos, la red neuronal devolvería 0010, ya que aparece dos veces frente a 0001 y 1000 que aparecen sólo una vez. Por tanto, el 1 está en la tercera posición (tercera columna de la Tabla 3) y eso indica que la acción realizada ha sido la de ver el video de tipo 2.

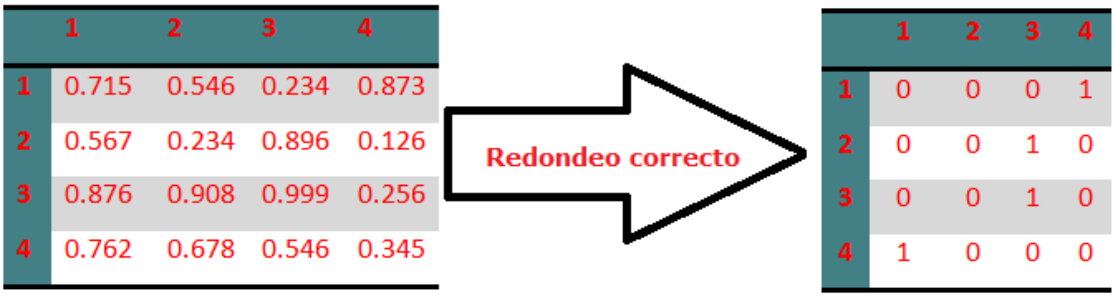


Ilustración 13. Redondeo correcto

4.2 Otra forma de entender los datos

Otro punto de vista de abordar los datos a clasificar es *clusterizar* los datos y ver cuántos tipos de movimientos de ojos hay. El *cluster* devolverá un número de movimientos, y será ese número el que determine cuántas clases se ha de tener.

De esta forma en lugar de tener cuatro clases distintas, una por cada actividad, se tendrán tantas clases como tipos de movimiento de los ojos haya. De esta manera, se puede hacer una clasificación respecto al tipo de movimiento de los ojos distinguiendo entre: un trazado que vaya siguiendo una línea recta, trazados con formas circulares, líneas verticales, etc.

4.3 Series Temporales

¿Qué es una serie temporal? Una serie temporal es una secuencia de datos, observaciones o valores, medidos en determinados momentos y ordenados cronológicamente.

Uno de los usos más habituales de las series de datos temporales es su análisis para predicción y pronóstico, así se hace por ejemplo con los datos climáticos, las acciones de bolsa, o las series de datos demográficos. La temporalidad de determinados eventos de las series es lo que normalmente se quiere caracterizar para resolver problemas en una amplia variedad de aplicaciones.

En este trabajo se usarán las series temporales para predecir las acciones que realiza un usuario frente a un ordenador.

Para llevar a cabo esto, lo primero que se ha hecho ha sido crear un programa en C que genere una serie temporal a partir de los datos generados por el *Eye Tracker* (cámara *Tobii*). Mediante el acceso mediante las funciones de la librería de la cámara, se obtienen los datos de la estimación de las coordenadas de la pantalla donde mira el ojo. Estos datos tienen el aspecto que se muestra en la Ilustración 14. Los primeros

datos corresponden a la información de la calibración, la primera columna es el tiempo y las dos siguientes columnas las coordenadas X e Y de la posición estimada del lugar de la pantalla donde la persona está mirando, necesarias para poder llevar a cabo el trabajo. Para la clasificación de las series temporales, se eliminarán los primeros datos y la primera columna, quedando así, las columnas de las coordenadas, además de acortar los ficheros para que todos tengan el mismo tamaño, en este caso, 800 líneas (que corresponden a 18 segundos aproximadamente). A partir de estos datos se generará un nuevo fichero que contiene la serie temporal con sus respectivas clases asignada según la codificación como muestran la Ilustración 15 y la Ilustración 16, con el objetivo de construir los datos de entrenamiento.

Después de haber hecho eso con todos los ficheros generados, se unifican en un único archivo para pasárselo a la red neuronal y que ésta pueda entrenar con él. Una vez entrenada se pasará otro fichero de test generado (con datos que no se encuentran en el fichero de entrenamiento), y que tienen las mismas características de formato que el último generado, es decir, una serie temporal, pero esta vez sin las clases, ya que se trata de saber a qué clase corresponde el fichero a testear (Ilustración 17).

```
p
Found
X230W-010103012143
X230W-010103012143
X2-30_Wide
X2-30_Wide
G5
0.9.3.22929.20130308.1721.root
ok
ipv4://172.28.195.1:(4455,4457)
----------------
ET created
Tracking started
Exception during ChannelHandlerFunctor.__call__(): 'int' object has no attribute 'x'
1167609917349000 626 624
1167609917394000 573 560
1167609917430000 625 643
1167609917466000 495 557
1167609917504000 623 651
1167609917541000 467 688
1167609917579000 604 714
1167609917616000 551 713
1167609917654000 634 736
1167609917694000 561 699
1167609917731000 603 722
1167609917769000 437 644
1167609917806000 608 714
```

Ilustración 14. Datos generados por la cámara al realizar una navegación

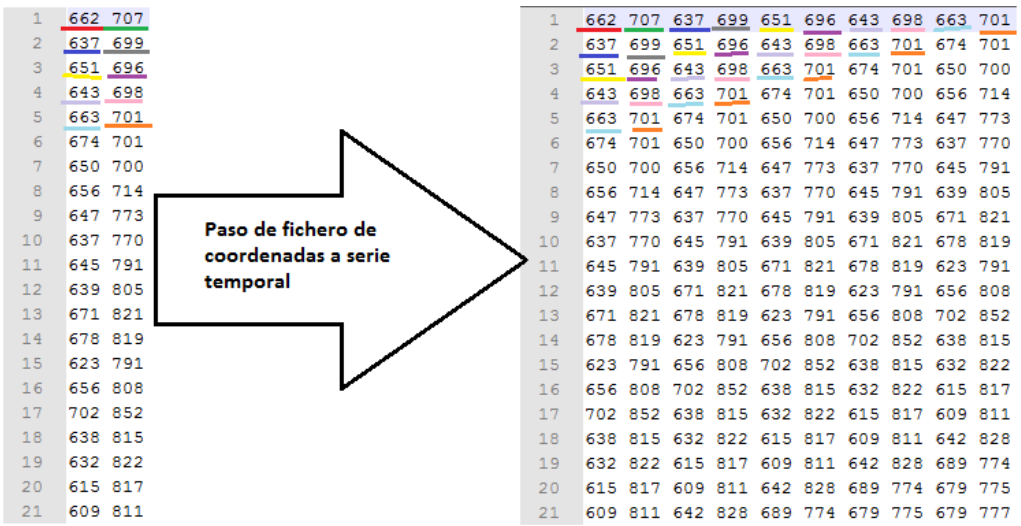


Ilustración 15. Conversión de datos a serie temporal con coordenadas retrasadas

Ilustración 16. Serie temporal con clase a la que pertenece

```
539 474 560 492 541 495 601 447 644 458
560 492 541 495 601 447 644 458 657 443
541 495 601 447 644 458 657 443 649 464
601 447 644 458 657 443 649 464 638 461
644 458 657 443 649 464 638 461 691 457
657 443 649 464 638 461 691 457 691 457
649 464 638 461 691 457 691 457 733 483
638 461 691 457 691 457 733 483 740 454
691 457 691 457 733 483 740 454 722 446
691 457 733 483 740 454 722 446 734 481
733 483 740 454 722 446 734 481 736 455
740 454 722 446 734 481 736 455 723 437
722 446 734 481 736 455 723 437 734 447
734 481 736 455 723 437 734 447 730 501
736 455 723 437 734 447 730 501 719 462
723 437 734 447 730 501 719 462 754 470
734 447 730 501 719 462 754 470 732 485
730 501 719 462 754 470 732 485 724 431
```

Ilustración 17. Fichero para realizar la explotación con la red neuronal

4.4 Codificación de la entrada

Para tener pruebas suficientes de que la clasificación se hace adecuadamente, se han variado el número de puntos tomados, haciendo primero que la red entrene con 5 puntos de la serie temporal y después con 25.

4.4.1 Diez neuronas en la capa de entrada

En un primer caso, para generar los nuevos ficheros explicados en el apartado anterior, se han cogido 5 puntos de coordenadas (X e Y), por tanto cada fila del fichero tendrá 10 columnas como mostraba la Ilustración 16. De este modo, la red neuronal tendrá 10 neuronas en la capa de entrada y 2 neuronas en la capa de salida como se había explicado anteriormente y una capa oculta con 20 neuronas donde se explicará el por qué, más adelante, quedando la red neuronal como se ve en la Ilustración 18.

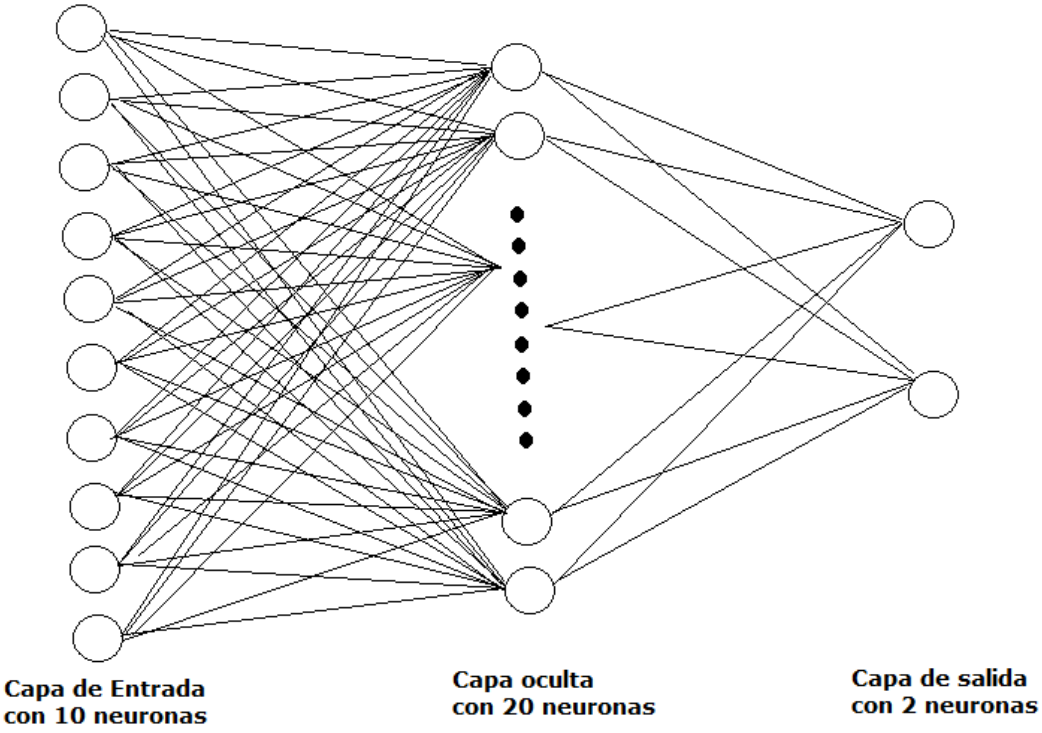


Ilustración 18. Red neuronal

4.4.2 Cincuenta neuronas en la capa de entrada

En un segundo caso, se ha probado con una serie más larga, cogiendo 25 puntos. Por tanto, cada fila del fichero tendrá 50 columnas. De este modo, la red neuronal tendrá 50 neuronas en la capa de entrada, 2 neuronas en la capa de salida como se había explicado anteriormente y una capa oculta con 20 neuronas. El aspecto es el mismo que la Ilustración 18 pero con 50 neuronas en la capa de entrada.

5 PRUEBAS Y RESULTADOS

Para detectar la actividad que ha estado realizando un usuario delante de la computadora, se llevarán a cabo dos tipos de análisis: inspección visual de las series temporales y pruebas mediante clasificación con una red neuronal.

5.1 Inspección visual de la acción realizada por un usuario a partir de los datos registrados por el *eye tracker*

De la inspección visual de las series temporales registradas se puede observar que no siempre se pueden distinguir bien entre movimientos oculares correspondientes a actividades similares como ocurre en el caso de los videos y la navegación. Por ejemplo, podría ponerse que en el primer vídeo, algo tranquilo y relajado sin apenas movimientos, el usuario fijara la mirada en un punto fijo ya que no tiene distracción alguna, y que en el otro vídeo al contener diversas escenas con diferentes movimientos, el sujeto haga distintos movimientos con el ojo y esto quede reflejado en el seguimiento ocular. Pero no es así, el resultado del seguimiento de los ojos durante estas dos tareas son similares, ya que en el primer video, la gran mayoría de los sujetos se dedican a mirar el paisaje y a observar cada punto de éste, lo que hace que las series temporales sean parecidas y difíciles de diferenciar a simple vista.

A continuación, se muestran dos imágenes que reflejan que el usuario ha estado leyendo (Ilustración 19 e Ilustración 20). Claramente se ve que se trata de esa actividad ya que los trazos van de izquierda a derecha y de arriba abajo.

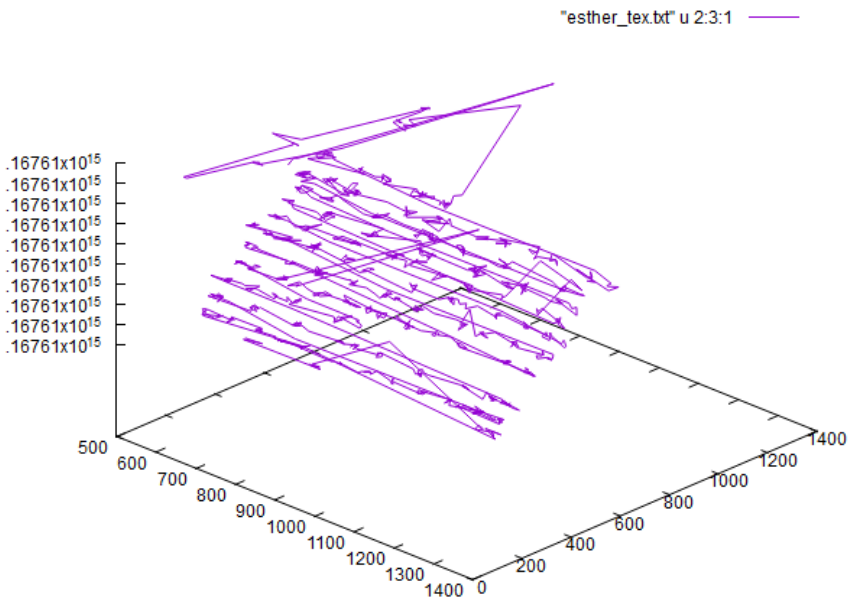


Ilustración 19. Ejemplo de serie temporal del seguimiento de ojos correspondiente a la acción de lectura

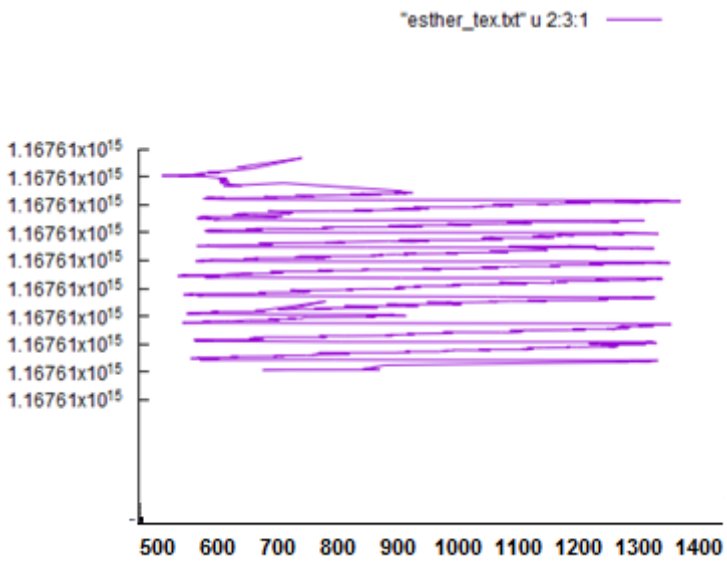


Ilustración 20. Proyección bidimensional de la gráfica anterior para su mejor visualización

En las cuatro siguientes gráficas se muestran las mismas actividades para usuarios distintos, donde se pueden observar las dificultades de diferenciar las acciones de navegar, ver un video de tipo 1 o de tipo 2.

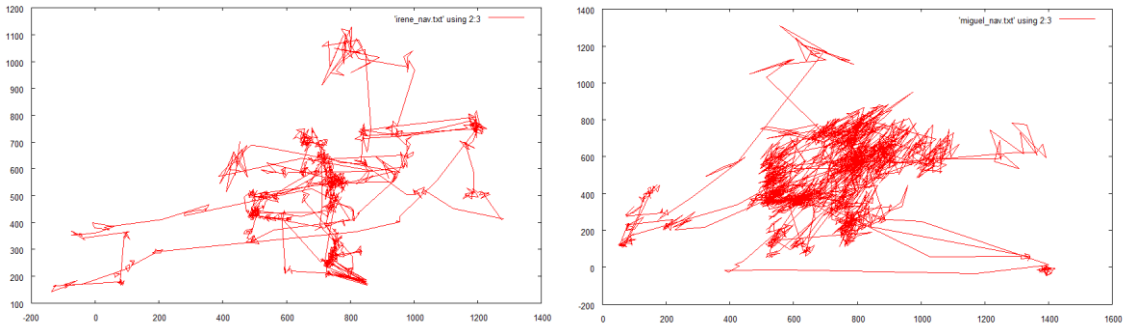


Ilustración 21. Seguimiento ocular durante la navegación en dos usuarios distintos

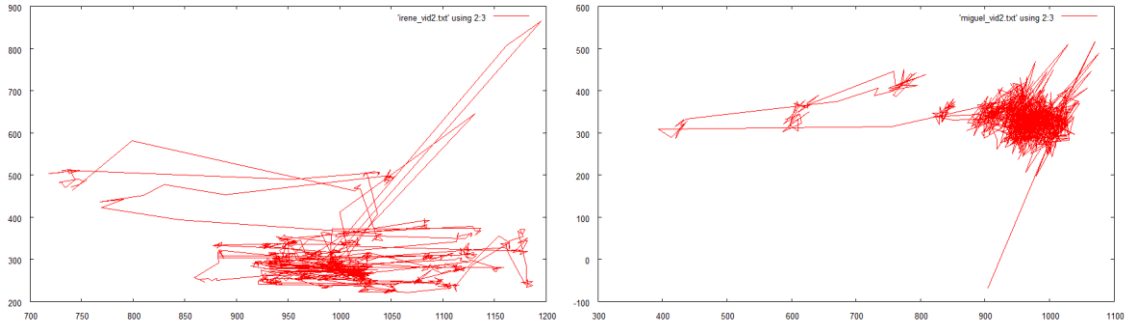


Ilustración 22. Seguimiento ocular durante el video 2 para dos usuarios distintos

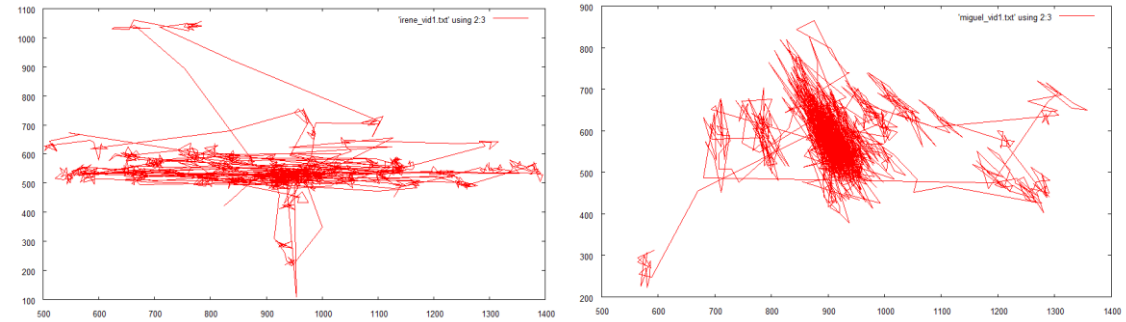


Ilustración 23. Seguimiento ocular durante el video 1 de dos usuarios distintos

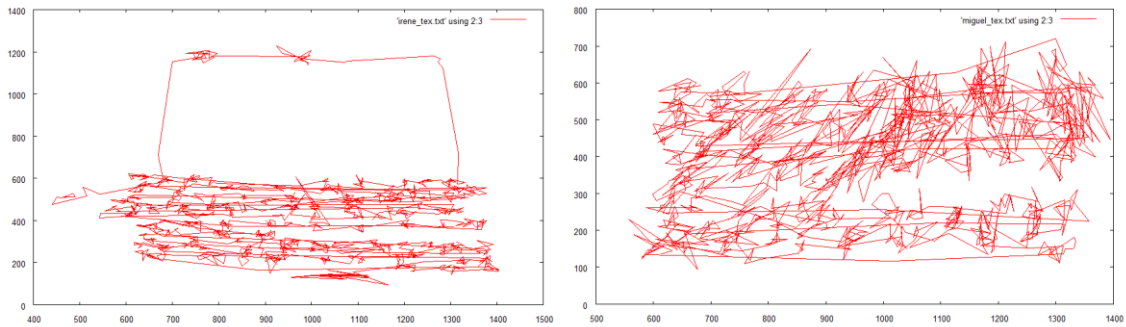


Ilustración 24. Seguimiento ocular durante una lectura de dos usuarios distintos

En el caso de los videos, es evidente que cada sujeto mirará de una forma distinta, observando lo que le sea más interesante. En cuanto a la navegación, habrá sujetos que conozcan la página por la que se navega y no necesiten mirar más allá ni buscar, mientras que los que la desconocen, tienden a observar más para encontrar el enlace indicado o el destino final. Todas estas diferencias, hacen que las pupilas del ojo, se desplacen en el tiempo de forma diferente. Sin embargo, el texto a leer, es algo que todo el mundo hacer por igual, quizás más rápidamente o menos, pero siempre de izquierda a derecha y de arriba abajo, dibujando siempre el mismo patrón con los ojos.

5.2 Detección de la acción realizada por un usuario mediante clasificación

En una aplicación que intente determinar el tipo de actividad que realiza una persona delante del ordenador, es necesaria una clasificación automática a partir de las series temporales generadas por la cámara. En esta sección se describe el análisis de la clasificación realizada con una red neuronal.

Estas pruebas se han llevado a cabo con la herramienta Matlab. Se ha comenzado creando dos variables: P, que contiene todo el conjunto de series temporales de cada prueba de cada sujeto y T, que contiene las clases de cada una de ellas (ver Ilustración 25).

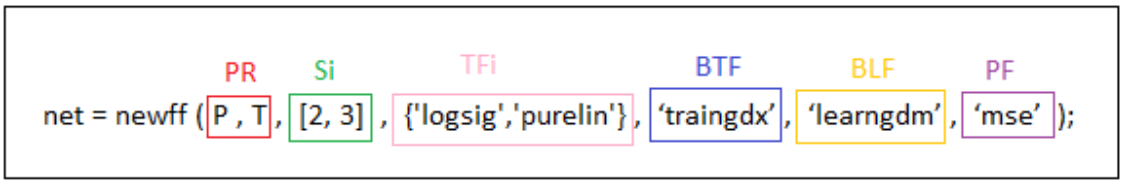
Una vez creadas las variables, crearemos la red neuronal de la siguiente forma:

```
net = newff ( PR, [S1, S2…SNi], {TF1, TF2…TFNi}, BTF,  BLF, PF)
```

- PR: R x 2 matriz de valores mínimos y máximos para R elementos de entrada.
- Si: Tamaño de la capa i, para Ni capas. Por ejemplo, si se tiene [2, 3] quiere decir que hay dos capas ocultas, una con 2 neuronas y la otra con 3.
- TFi: Función de transferencia de la capa i, por defecto = ‘tansig’.
- BTF: Función Backpropagation de entrenamiento de la red., por defecto = ‘traingdx’.

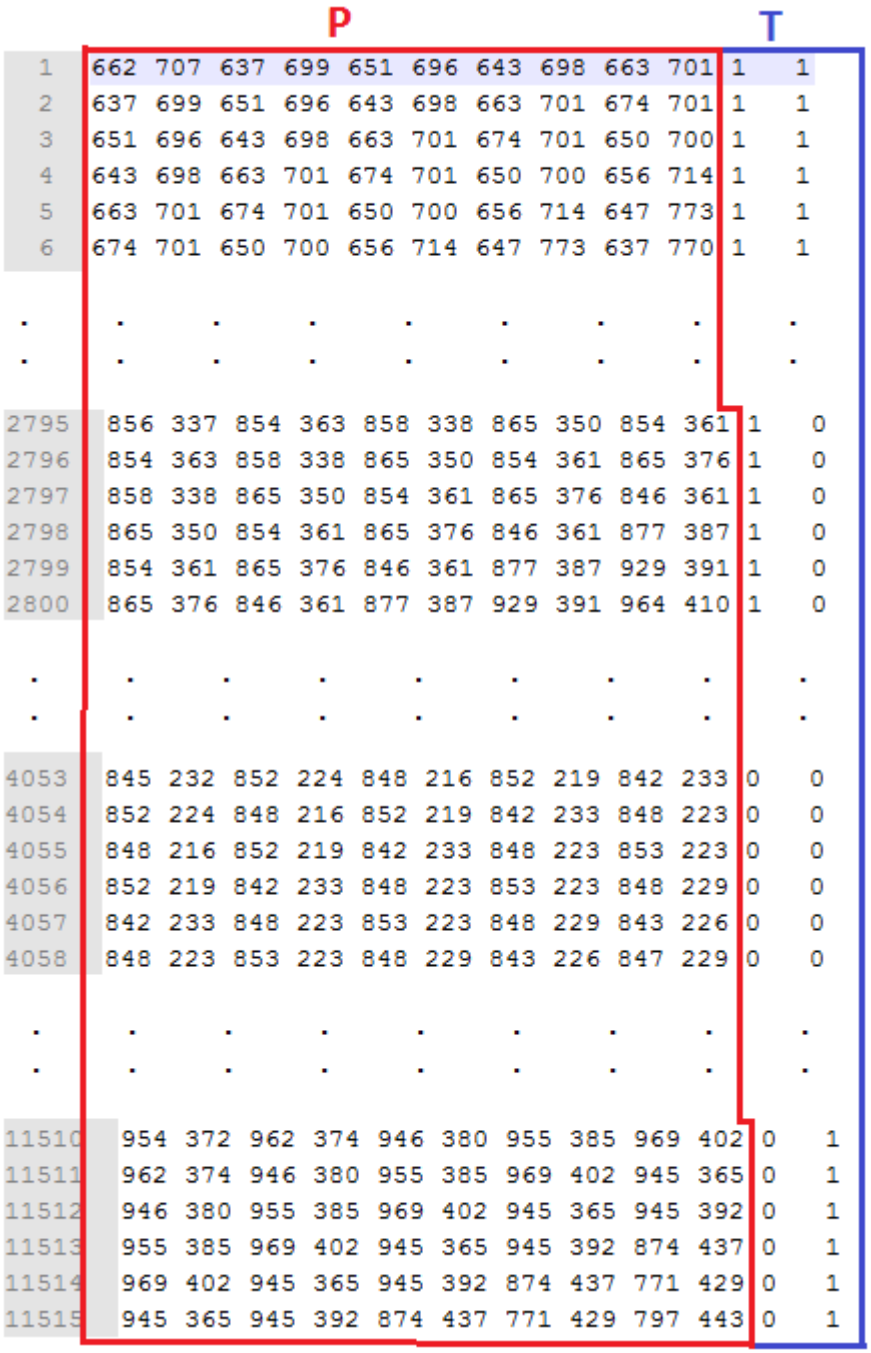
- BLF: Función de peso Backprogapation / sesgo de aprendizaje, por defecto = ‘learngdm’.
- PF: Función de rendimiento, por defecto = ‘mse’.

Para verlo más claro un ejemplo sería:



5.2.1 Detección con 5 puntos de entradas: 10 neuronas de entrada

En este caso, la función introducida ha sido la siguiente: *net = newff(P’, T’, 20);* sin necesidad de poner el resto de valores (tansig, traingdx, learngdm, learngdm) , ya que así, se toman los que vienen por defecto. De este modo, P son los datos de la serie temporal con los que la red neuronal entrenará, T la clase a la que pertenecen dichos datos y el 20, es el número de neuronas que tiene la capa oculta. Se ha decidido trabajar con una única capa oculta ya que los resultados han sido satisfactorios.



1	662	707	637	699	651	696	643	698	663	701	1	1
2	637	699	651	696	643	698	663	701	674	701	1	1
3	651	696	643	698	663	701	674	701	650	700	1	1
4	643	698	663	701	674	701	650	700	656	714	1	1
5	663	701	674	701	650	700	656	714	647	773	1	1
6	674	701	650	700	656	714	647	773	637	770	1	1
.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.
2795	856	337	854	363	858	338	865	350	854	361	1	0
2796	854	363	858	338	865	350	854	361	865	376	1	0
2797	858	338	865	350	854	361	865	376	846	361	1	0
2798	865	350	854	361	865	376	846	361	877	387	1	0
2799	854	361	865	376	846	361	877	387	929	391	1	0
2800	865	376	846	361	877	387	929	391	964	410	1	0
.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.
4053	845	232	852	224	848	216	852	219	842	233	0	0
4054	852	224	848	216	852	219	842	233	848	223	0	0
4055	848	216	852	219	842	233	848	223	853	223	0	0
4056	852	219	842	233	848	223	853	223	848	229	0	0
4057	842	233	848	223	853	223	848	229	843	226	0	0
4058	848	223	853	223	848	229	843	226	847	229	0	0
.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.
11510	954	372	962	374	946	380	955	385	969	402	0	1
11511	962	374	946	380	955	385	969	402	945	365	0	1
11512	946	380	955	385	969	402	945	365	945	392	0	1
11513	955	385	969	402	945	365	945	392	874	437	0	1
11514	969	402	945	365	945	392	874	437	771	429	0	1
11515	945	365	945	392	874	437	771	429	797	443	0	1

Ilustración 25. División de los datos para la red neuronal

Con esta instrucción, la red ya contiene los valores para trabajar. Ahora hay que entrenar: *net = train(net,P', T')*. Al ejecutar este comando, saldrá una ventana similar a la Ilustración 26, donde se puede observar cómo la red entrena y va aprendiendo.

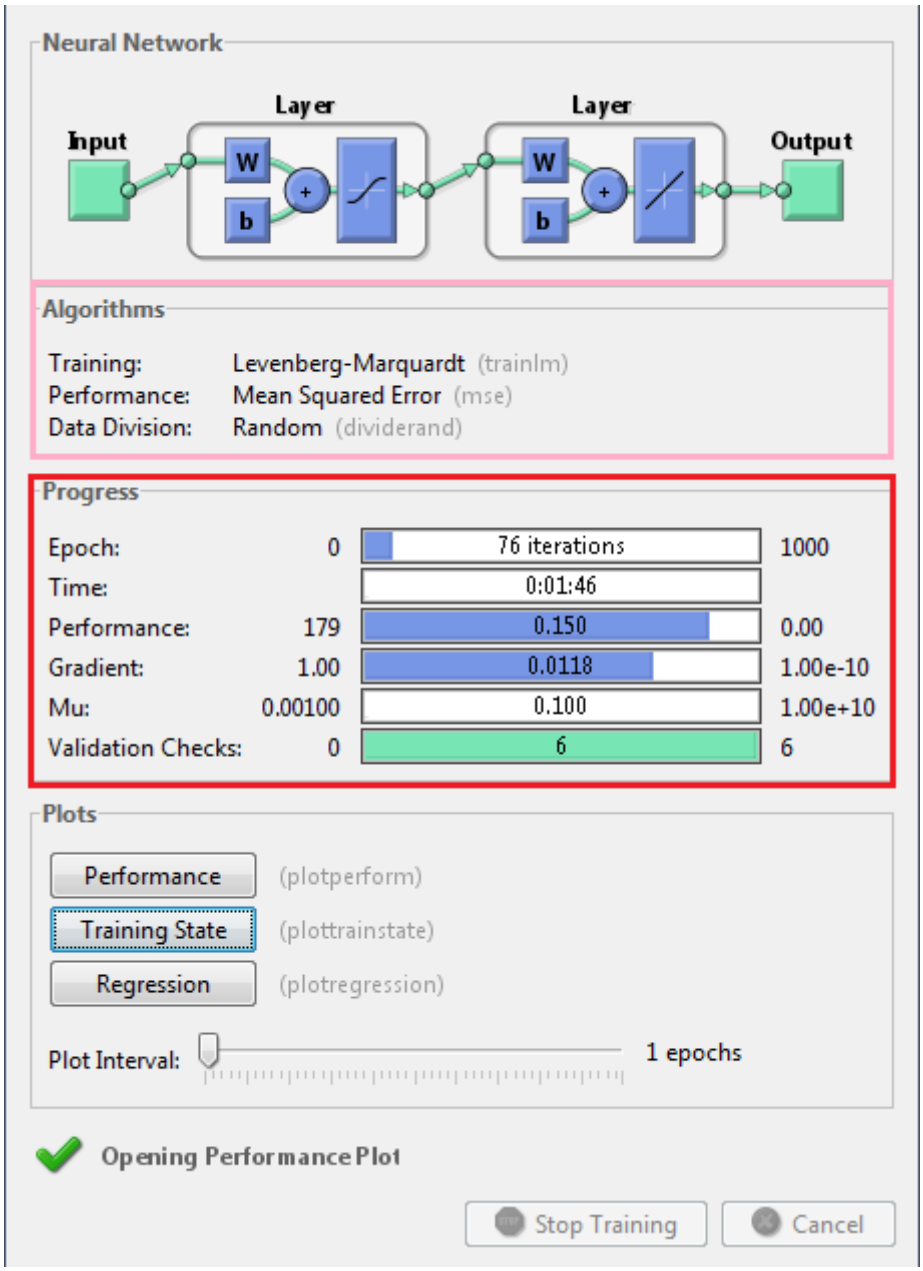


Ilustración 26. Entrenamiento de la red neuronal

En el recuadro marcado en rosa se ven las opciones escogidas por defecto como es el caso de la función de rendimiento puesta con 'mse'. Marcado en rojo, se observan una serie de datos:

- Épocas: Por defecto, si no se elige un número de épocas, el número máximo de épocas que realizará será de 1000. En este caso, la red neuronal ha realizado 76 iteraciones en el entrenamiento.

- Tiempo: La red neuronal ha tardado un tiempo de 1 minutos y 46 segundos en realizar el entrenamiento.

- Rendimiento: El rendimiento ha alcanzado un valor de 0.150, muy cercano a 0, lo que muestra que la red ha ido aprendiendo adecuadamente hasta conseguir error mínimo (ver Ilustración 28).

- Gradiente: El gradiente tiende a 0. Esto sucede cuando los pesos quedan ajustados, es decir, no se producen modificaciones en los pesos y el aprendizaje queda detenido.

- Mu: Se trata de una variable constante con valor de 0.01.

- Validación: La red neuronal coge un porcentaje mínimo de los datos para realizar la validación. El entrenamiento parará cuando en la validación, la red encuentre 6 veces seguidas un 100% de aciertos.

Para determinar cuándo se detendrá el proceso de aprendizaje, es necesario establecer una condición de parada: cuando el cálculo del error cuadrado sobre todos los ejemplos de entrenamiento ha alcanzado un mínimo, cuando para cada uno de los ejemplos dados, el error observado está por debajo de un determinado umbral o por ejemplo cuando un cierto número de ciclos y/o pasos de entrenamiento hayan sido completamente corridos. En este caso, que la red encuentre 6 veces consecutivas un 100% de aciertos es la condición de parada y por tanto, la red no tiene sobreaprendizaje.

Una vez finalizado el proceso de entrenamiento, se puede ver la recta de regresión, y la gráfica de rendimiento.

¿Qué es la recta de regresión y para qué sirve? La recta de regresión es la que mejor se ajusta a la nube de puntos. Sirve para hacer estimaciones teniendo en cuenta que la estimación es más fiable para los valores próximos a la media, que los valores

obtenidos son aproximaciones en términos de probabilidad (es probable que el valor x4, sea y4), que la fiabilidad aumenta al aumentar el número de datos y que ésta fiabilidad es mayor cuanto más fuerte sea la correlación, donde por correlación se entiende que la nube de puntos se parezca a la recta.

La nube de puntos obtenida en este caso con el entrenamiento realizado ha sido la que muestra la Ilustración 27. En todos los casos (entrenamiento, test y validación), los puntos están tan juntos que lo único que se aprecia es una recta gruesa vertical, debido a que dichos puntos se agrupan en los números 0 y 1 del eje X a lo largo del eje Y, ya que lo que se busca es un resultado binario de ese tipo.

Decir también que no hay una correlación lineal, ya que los puntos no se distribuyen alrededor de la recta, además se trata de una correlación positiva o directa, pues la recta de regresión en todos los casos es creciente (al aumentar una variable, la otra también tiene tendencia a aumentar).

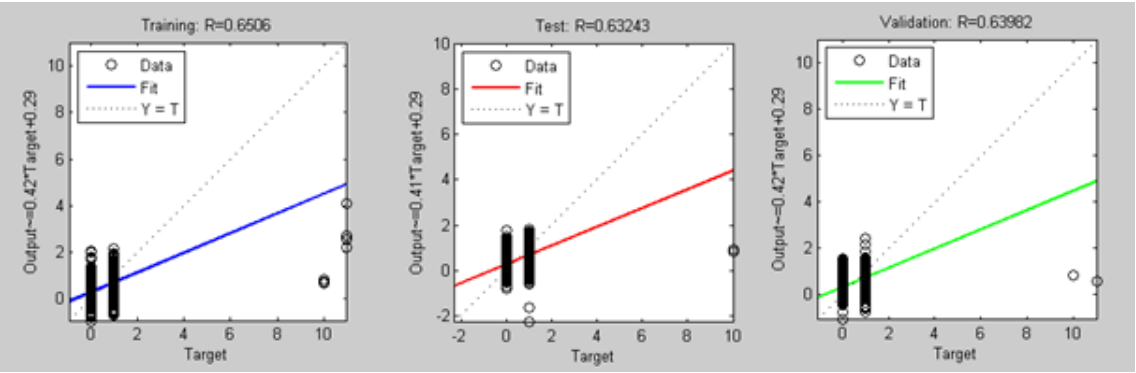


Ilustración 27. Rectas de regresión

El otro aspecto a analizar de la red neuronal es el rendimiento. La Ilustración 28 muestra una gráfica en la que se puede observar que conforme pasan las épocas y la red va entrenando, el error cuadrático medio es cada vez menor, teniendo un límite casi de 0 y consiguiendo una estabilidad, lo que indica que la red neuronal aprende adecuadamente.

$$\frac{1}{n}\sum_{i=1}^{n}(¥_i - Y_i)$$

Ecuación 2. Error Cuadrático Medio, donde ¥ es un vector de n predicciones e Y es el vector de los verdaderos valores

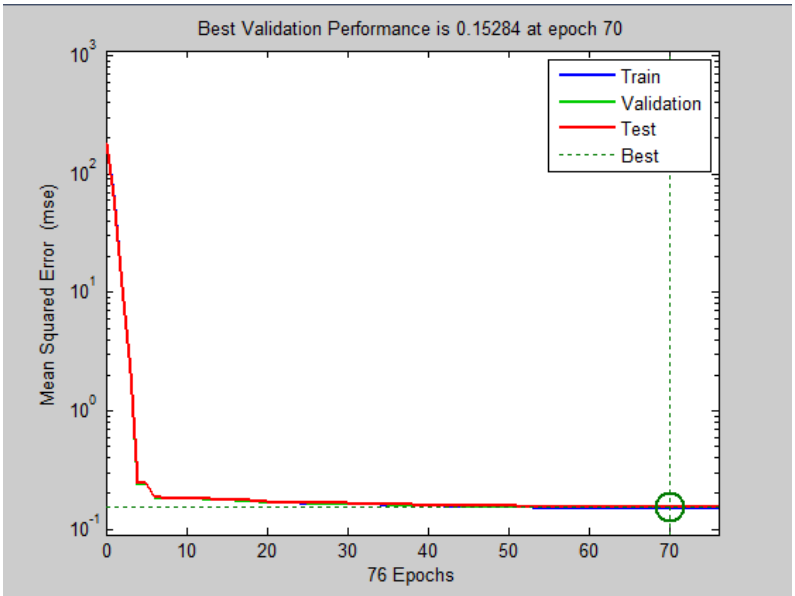


Ilustración 28. Gráfica de rendimiento de la red neuronal. Error Cuadrático Medio vs Épocas

Tras haber entrenado la red neuronal con un número considerable de sujetos, los cuales realizaron las cuatro pruebas por igual, se eligieron cinco nuevos voluntarios para que realizaran las mismas pruebas. Los cuatro ficheros generados de las distintas pruebas (leer, ver video1, ver video2, navegar), siguieron el mismo proceso que los anteriores, hasta convertirlos en una serie temporal sin clases.

Es ahora cuando la red neuronal va a tener que ser capaz de clasificar un nuevo fichero sin clases con lo aprendido anteriormente.

Para comprobar que es capaz de clasificar cualquier actividad realizada la RNA tendrá que ser capaz de clasificar las 20 actividades en su grupo de clase correspondiente.

Para hacer que la red pase al modo test, basta con poner el comando: *res=sim(net, datos_a_clasificar')'.* En la variable "res", se guardará la clasificación de cada serie temporal del fichero. Se trata de una matriz de tantas filas como tenga el fichero y dos columnas. Como se explicó anteriormente, lo que genera son valores cercanos a 1 y 0, como se puede observar en la Ilustración 29. Lo que se necesita ahora, es obtener valores exactos a 0 y 1 (Ilustración 30), por lo que para redondear estos datos obtenidos se hará uso de la sentencia*: y=round(res).*

Por último, hay que contar qué valor de los cuatro aparece más veces (00, 01, 10, o 11). Para esto, hayamos la moda con la siguiente sentencia: *mode(y);*

	1	2
1	0.7597	0.1389
2	0.7571	0.1101
3	0.7783	0.1025
4	0.7951	0.0707
5	0.8085	0.0852
6	0.8221	0.0893
7	0.8018	0.0714
8	0.8131	0.0885
9	0.8190	0.1106
10	0.8427	0.1151
11	0.8418	0.1102
12	0.8157	0.0860
13	0.7944	0.0721
14	0.7939	0.0651
15	0.7907	0.0351
16	0.7919	-0.0183
17	0.7894	-0.0524
18	0.7644	-0.0656
19	0.7291	-0.0715
20	0.8080	-0.0340
21	0.8524	-0.0070
22	0.8126	-0.0161
23	0.8373	0.0016
24	0.8761	-0.0123
25	0.8662	-0.0131
26	0.8472	-0.0257
27	0.8692	-0.0113

Ilustración 29. Matriz con la clasificación devuelta para cada línea del fichero

y <796x2 double>

	1	2
1	1	0
2	1	0
3	1	0
4	1	0
5	1	0
6	1	0
7	1	0
8	1	0
9	1	0
10	1	0
11	1	0
12	1	0
13	1	0
14	1	0

Ilustración 30. Valores redondeados

Todas estas instrucciones se pueden resumir en una sola sentencia: mode(round(sim(net,datos_a_clasificar')')); como muestra la tercera línea de la Ilustración 31. Esa misma imagen, muestra la clasificación que ha hecho la red neuronal con los datos de cada actividad del usuario Borja. En primer lugar, se le pasó un fichero cuyos datos habían sido generados por dicho sujeto mientras se dedicaba a leer. La clase de la lectura es la "00", por tanto la red neuronal tiene que devolver en la variable "ans" 00. Lo mismo para el resto de actividades: 01 si se trata del video 1, 10 para el video 2 y 11 para la navegación.

A continuación, se muestran cinco ilustraciones (una por cada usuario) que reflejan la clasificación de cinco sujetos diferentes habiendo realizado las cuatro pruebas por igual. ¿Será la RNA capaz de clasificar correctamente la actividad de cada participante?

```
>> net = newff(P',T',20);
>> net = train(net, P', T');
>> mode(round(sim(net,borja_lee')'))

ans =

     0     0

>> mode(round(sim(net,borja_video1')'))

ans =

     0     1

>> mode(round(sim(net,borja_video2')'))

ans =

     1     0

>> mode(round(sim(net,borja_navega')'))

ans =

     1     1
```

Clase 0 0 = leer texto

Clase 0 1 = ver video 1

Clase 1 0 = ver video 2

Clase 1 1 = navegar

Ilustración 31. Clasificación de cada actividad del sujeto Borja

```
>> mode(round(sim(net,gonzalo_lee')'))

ans =

     0     0

>> mode(round(sim(net,gonzalo_video1')'))

ans =

     0     1

>> mode(round(sim(net,gonzalo_video2')'))

ans =

     1     0

>> mode(round(sim(net,gonzalo_navega')'))

ans =

     1     1
```

Ilustración 32. Clasificación de cada actividad del sujeto Gonzalo

```
>> mode(round(sim(net,laura_lee')'))

ans =

     0     0

>> mode(round(sim(net,laura_video1')'))

ans =

     0     1

>> mode(round(sim(net,laura_video2')'))

ans =

     1     0

>> mode(round(sim(net,laura_navega')'))

ans =

     1     1
```

Ilustración 33. Clasificación de cada actividad del sujeto Laura

```
>> mode(round(sim(net,pablo_lee')'))

ans =

     0     0

>> mode(round(sim(net,pablo_video1')'))

ans =

     0     1

>> mode(round(sim(net,pablo_video2')'))

ans =

     1     0

>> mode(round(sim(net,pablo_navega')'))

ans =

     1     1
```

Ilustración 34. Clasificación de cada actividad del sujeto Pablo

```
>> mode(round(sim(net,esther_lee')'))

ans =

     0     0

>> mode(round(sim(net,esther_video1')'))

ans =

     0     1

>> mode(round(sim(net,esther_video2')'))

ans =

     1     0

>> mode(round(sim(net,esther_navega')'))

ans =

     1     1
```

Ilustración 35. Clasificación de cada actividad del sujeto Esther

Como se ha observado en las imágenes anteriores, en todos los casos, la red neuronal ha sido capaz de reconocer cada actividad, con un porcentaje del 100% de aciertos. Por tanto, es capaz de clasificarlos, pero no solo eso, es capaz de clasificarlos correctamente.

5.2.2 Detección con 25 puntos de entradas: 50 neuronas de entrada

Para este caso, la función introducida ha sido la misma que anteriormente: *net = newff(P', T', 20);* sin necesidad de poner el resto de valores (tansig, traingdx, learngdm, learngdm) , ya que así, se toman los que vienen por defecto. De este modo, P son los datos de la serie temporal con los que la red neuronal entrenará pero esta vez con 50 columnas en lugar de 10, T la clase a la que pertenecen dichos datos y el 20, es el número de neuronas que tiene la capa oculta. Se ha decidido trabajar con una única capa oculta ya que los resultados han sido satisfactorios.

Con esta instrucción, la red ya contiene los valores para trabajar. Ahora hay que entrenar igual que se hizo anteriormente: *net = train(net,P', T')*. Al ejecutar este

comando, saldrá una ventana similar a la Ilustración 36, donde se puede observar cómo la red entrena y va aprendiendo.

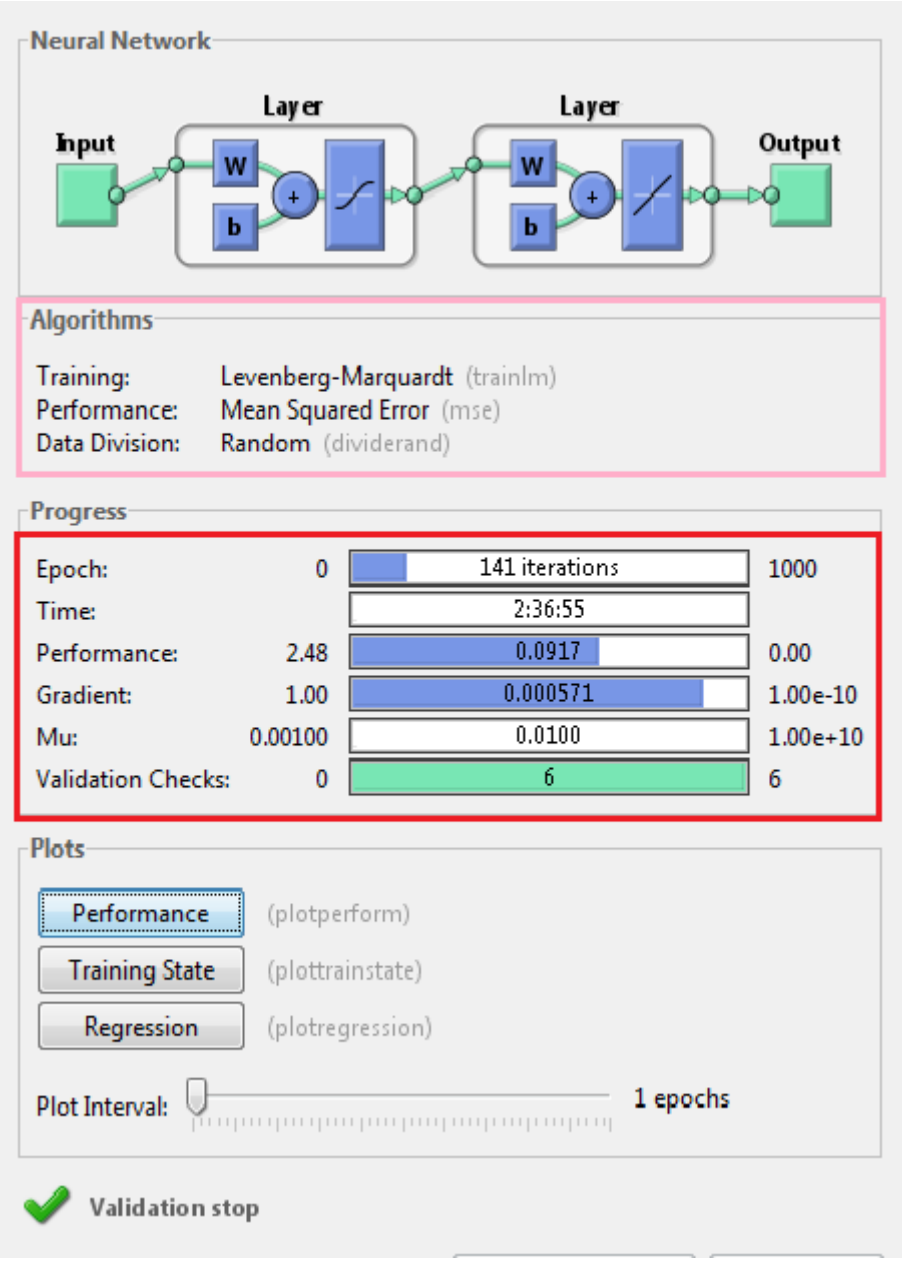


Ilustración 36. Entrenamiento de la red neuronal

En el recuadro marcado en rosa se ven las opciones escogidas por defecto como es el caso de la función de rendimiento puesta con 'mse'. Marcado en rojo, se observan una serie de datos:

- Épocas: Por defecto, si no se elige un número de épocas, el número máximo de épocas que realizará será de 1000. En este caso, la red neuronal ha realizado 141 épocas en el entrenamiento.

- Tiempo: La red neuronal ha tardado un tiempo de 2 horas y 36 minutos (debido a que la máquina en la que se probó no era demasiado potente, en cambio en una máquina más potente para una red con 200 entradas se tardó unos 9 minutos) en realizar el entrenamiento.

- Rendimiento: El rendimiento ha alcanzado un valor de 0.0917, muy cercano a 0, lo que muestra que la red ha ido aprendiendo adecuadamente hasta conseguir error mínimo (ver Ilustración 37).

- Gradiente: El gradiente tiende a 0. Esto sucede cuando los pesos quedan ajustados, es decir, no se producen modificaciones en los pesos y el aprendizaje queda detenido.

- Mu: Se trata de una variable constante con valor de 0.01.

- Validación: La red neuronal coge un porcentaje mínimo de los datos para realizar la validación. El entrenamiento parará cuando en la validación, la red encuentre 6 veces seguidas un 100% de aciertos.

Una vez finalizado el proceso de entrenamiento, se observará qué ocurre con la gráfica de rendimiento y el estado del entrenamiento.

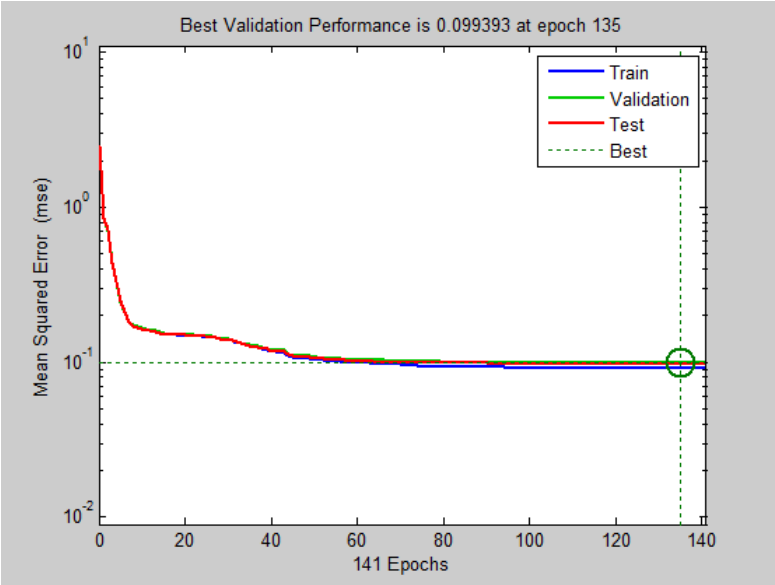


Ilustración 37. Rendimiento: Épocas versus ECM

El rendimiento que se ha obtenido se muestra en la Ilustración 37. Se observa que conforme pasan las épocas y la red va entrenando, el error cuadrático medio es cada vez menor, teniendo un límite casi de 0 y consiguiendo una estabilidad, lo que indica que la red neuronal aprende adecuadamente.

La siguiente imagen muestra el estado del entrenamiento, donde se observa que la red no ha tenido sobreaprendizaje, ya que la validación alcanza su valor máximo cuando el gradiente tiende a 0.

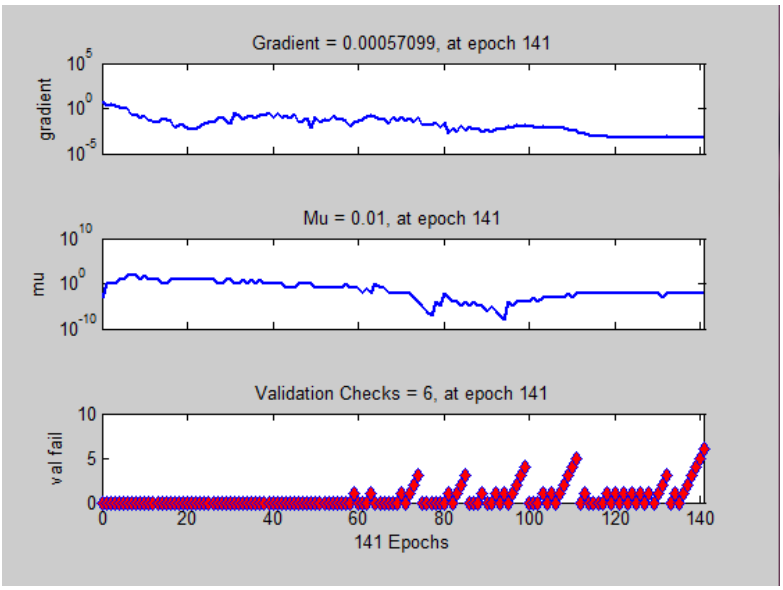


Ilustración 38. Estado del entrenamiento

A continuación, se muestran cinco ilustraciones (una por cada usuario) que reflejan la clasificación de cinco sujetos diferentes habiendo realizado las cuatro pruebas por igual. ¿Será ahora la RNA capaz de clasificar correctamente la actividad de cada participante?

```
>> mode(round(sim(net,borja_lee(:,1:50)')'))

ans =

     0     0

>> mode(round(sim(net,borja_video1(:,1:50)')'))

ans =

     0     1

>> mode(round(sim(net,borja_video2(:,1:50)')'))

ans =

     1     0

>> mode(round(sim(net,borja_navega(:,1:50)')'))

ans =

     1     1
```

Clase 0 0 = leer

Clase 0 1 = video 1

Clase 1 0 = video 2

Clase 1 1 = navegar

Ilustración 39. Clasificación de cada actividad del sujeto Borja

```
>> mode(round (sim(net, gallas_lee(:,1:50)')'))

ans =

     0     0

>> mode(round (sim(net, gallas_video1(:,1:50)')'))

ans =

     0     1

>> mode(round (sim(net, gallas_video2(:,1:50)')'))

ans =

     1     0

>> mode(round (sim(net, gallas_navega(:,1:50)')'))

ans =

     1     1
```

Ilustración 40. Clasificación de cada actividad del sujeto Gallas

```
>> mode(round (sim(net, laura_lee(:,1:50)')'))

ans =

     0     0

>> mode(round (sim(net, laura_video1(:,1:50)')'))

ans =

     0     1

>> mode(round (sim(net, laura_video2(:,1:50)')'))

ans =

     1     0

>> mode(round (sim(net, laura_navega(:,1:50)')'))

ans =

     1     1
```

Ilustración 41. Clasificación de cada actividad del sujeto Laura

```
>> mode(round (sim(net, esther_lee(:,1:50)')'))

ans =

     0     0

>> mode(round (sim(net, esther_video1(:,1:50)')'))

ans =

     0     1

>> mode(round (sim(net, esther_video2(:,1:50)')'))

ans =

     1     0

>> mode(round (sim(net, esther_navega(:,1:50)')'))

ans =

     1     1
```

Ilustración 42. Clasificación de cada actividad del sujeto Esther

```
>> mode(round (sim(net, pablo_lee(:,1:50)')'))

ans =

     0     0

>> mode(round (sim(net, pablo_video1(:,1:50)')'))

ans =

     0     1

>> mode(round (sim(net, pablo_video2(:,1:50)')'))

ans =

     1     0

>> mode(round (sim(net, pablo_nav(:,1:50)')'))

ans =

     1     1
```

Ilustración 43. Clasificación de cada actividad del sujeto Pablo

Como muestran las imágenes anteriores, la red neuronal consigue detectar las actividades correctamente que han realizado cada uno de los voluntarios. Pero, ¿Qué ocurre cuando se pasan unos datos que no pertenecen a ninguna de las clases anteriores? Lo que sucede es que la red intenta buscar una similitud, es decir, devolverá una de las cuatro clases, siendo ésta la que más parecido tenga con alguna de las actividades. Por tanto, para saber si se está en un caso como este, se hallará el valor de un indicador con el que se sabrá si la clase clasificada es realmente la que debe ser o se trata de una similitud.

Para esto, se mirará la salida resultante de la clasificación, contando la cantidad de veces que aparece cada clase en esa salida. Por ejemplo, si volvemos a la Ilustración 11 se tiene que la clase 0 0 aparece ocho veces, la clase 0 1 aparece seis veces y que la clase 1 1 se muestra una vez. Por tanto, según esto, la clase ganadora es la clase 0 0 porque es la que más veces aparece. Para cerciorarnos de esto, hallaremos el indicador de cada clase: $indicadorClaseX = nº\ vecesClaseX / \sum nº\ vecesRestoClases$

Indicador00 = 8/(6+1+0) = 1.14

Indicador01 = 6/(8+1+0) = 0.66

Indicador10 = 0/(8+6+1) = 0

Indicador11 = 1/(8+6+0) = 0.07

De este modo, si hay algún valor que sobrepase de 1, se puede asegurar que la clasificación es correcta, ya que como se verá más adelante, si todos los indicadores tienen un valor cercano a 0, se trata de un caso en el que la red neuronal ha realizado una similitud y la clasificación es errónea.

A continuación se muestra la comprobación para el sujeto Borja para las cuatro actividades, mostrando que la clasificación es fiable debido al indicador descrito.

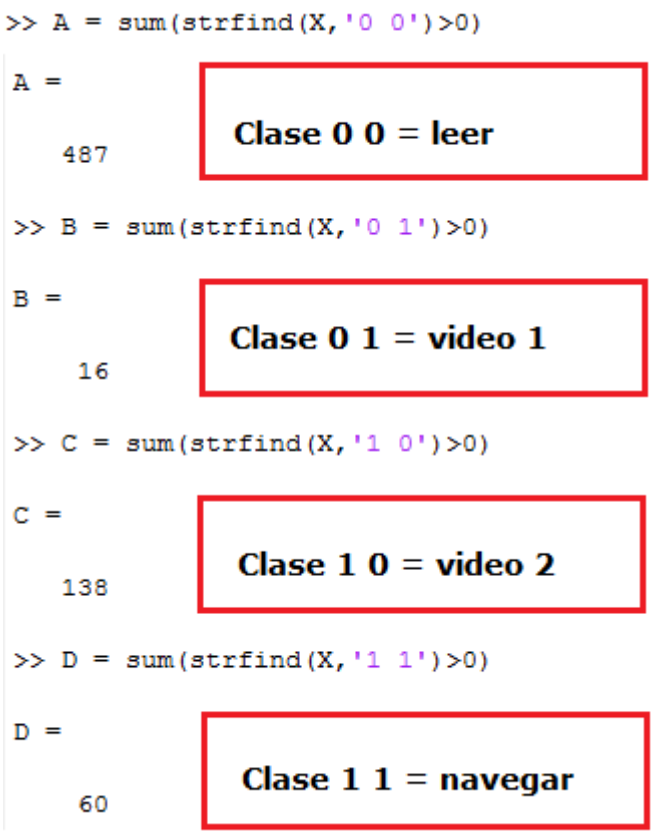


Ilustración 44. Cantidad de apariciones de cada clase para la actividad de lectura realizada por el sujeto Borja

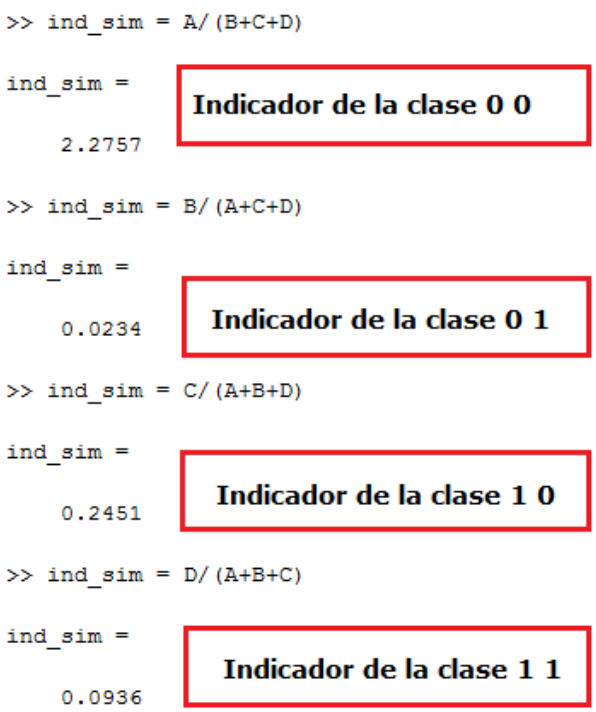


Ilustración 45. Indicadores de cada clase para la actividad de lectura del sujeto Borja

Como muestra la Ilustración 44, la clase 0 0 aparece 487 veces y es la elegida en la clasificación de los datos. Al hallar el indicador de cada clase, se ve que para la clase 0 0, alcanza un valor de 2.27, mientras que el resto están cercanos a 0. Esto indica que la clasificación es correcta y no es una similitud.

Para clarificar lo explicado, se muestran otro ejemplo de la actividad del video 2 realizada por el sujeto Borja. Esta actividad corresponde a la clase 1 0, y como se observa en la Ilustración 46, se clasifica correctamente con una aparición de la clase de 696 veces.

```
>> M = round (sim(net, borja_video2(:,1:50)'));
>> X = mat2str(abs(M))

X =

[1 0;1 0;1 0;1 0;1 0;1 0;1 0;1 0;1 0;1 0;1 0;1 0;1 0;1 0;1 0;1 0;1 0;1 0;1 0;1 0;1 0;1 0;1 0;1 (

>> A = sum(strfind(X,'0 0')>0)

A =

     5

>> B = sum(strfind(X,'0 1')>0)

B =

     0

>> C = sum(strfind(X,'1 0')>0)

C =

   696

>> D = sum(strfind(X,'1 1')>0)

D =

     0
```

Ilustración 46. Cantidad de veces que aparece cada clase en la actividad de ver un video de tipo 2 por el sujeto Borja

```
>> ind_sim = A/(B+C+D)

ind_sim =

    0.0072

>> ind_sim = B/(A+C+D)

ind_sim =

     0

>> ind_sim = C/(A+B+D)

ind_sim =

  139.2000

>> ind_sim = D/(A+B+C)

ind_sim =

     0
```

Ilustración 47. Indicador de cada clase en la actividad del video 2 realizada por el sujeto Borja

Como se puede observar, esta clasificación también es correcta ya que el indicador de la clase ganadora sobrepasa muy por encima el valor 1, mientras que el resto de indicadores de las clases restantes, se quedan en 0 o cerca de éste.

Ahora se mostrará un caso distinto, en el que los datos que se pasarán a la red neuronal no son ninguna de las cuatro actividades citadas. Los resultados esperados son que clasifique los datos en alguna de las cuatro clases y que el indicador se acerque a 0 para todas éstas.

En la Ilustración 48 se ve como se pasa un nuevo fichero a clasificar, el cual no pertenece a ninguna de las cuatro actividades y el resultado de la clasificación es la clase 1 0 (video 2).

```
>> mezclado=csvread('mezclado.csv');
>> mode(round (sim(net, mezclado(:,1:50)')'))

ans =

     1     0
```

Ilustración 48. Clasificación de datos ajenos a las cuatro clases

En la siguiente Ilustración 49, se obtienen la cantidad de veces que se asemejan los datos a las cuatro clases, siendo este valor bastante equitativo entre las cuatro.

```
>> M = round (sim(net, mezclado(:,1:50)'');
>> X = mat2str(abs(M))

>> A = sum(strfind(X,'0 0')>0)

A =

   219

>> B = sum(strfind(X,'0 1')>0)

B =

   102

>> C = sum(strfind(X,'1 0')>0)

C =

   244

>> D = sum(strfind(X,'1 1')>0)

D =

   135
```

Ilustración 49. Cantidad de veces que se asemejan los datos a las cuatro clases

Finalmente, en la Ilustración 50 se han calculado los indicadores para cada clase, teniendo todos ellos un valor cercano a 0. Esto quiere decir, que ninguna de las cuatro clases es la correcta, que la clasificación no es correcta y los datos no pertenecen a ninguna de las cuatro clases, y por tanto, no se ha realizado ninguna de las cuatro actividades.

```
>> ind_sim00 = A/(B+C+D)

ind_sim00 =

    0.4553

>> ind_sim01 = B/(A+C+D)

ind_sim01 =

    0.1706

>> ind_sim10 = C/(A+B+D)

ind_sim10 =

    0.5351

>> ind_sim11 = D/(A+B+C)

ind_sim11 =

    0.2389
```

Ilustración 50. Indicadores de cada clase de la nueva actividad realizada

6 CONCLUSIONES

La tecnología *Eye Tracking* o seguimiento de los ojos permite un abanico muy amplio de aplicaciones de interacción persona-ordenador. Por otro lado, gracias a los patrones que deja el ojo a lo largo de una trayectoria durante la realización de una actividad, se puede abordar el problema de determinar qué actividad realiza un sujeto frente a una pantalla de ordenador.

En este trabajo se ha proporcionado una visión de cómo entender los gestos de la mirada y se ha explicado una posible aplicación de los gestos oculares explicando cómo usar los datos de seguimiento ocular para tratar de clasificar acciones que realiza una persona frente a una computadora.

Un factor importante en este proceso, ha sido contar con la tecnología de seguimiento ocular *Tobii*. Esta tecnología es una de las más avanzadas, proporcionando una alta resolución y un calibrado de calidad, permitiendo la movilidad de la cabeza a los usuarios, algo que en otras tecnologías no es posible y resulta en un grave deterioro en la información registrada por la cámara.

El interés general para determinar actividad e intenciones a través de los ojos y la mirada va en ascenso y es el resultado del desarrollo de las técnicas de seguimiento ocular, basadas en vídeo-cámaras que hacen posible el registro de movimientos oculares en tiempo real. De esta manera los ojos también se pueden utilizar como elementos de monitorización y control, aumentando así el *ancho de banda* de la comunicación desde el usuario a la computadora.

Por último decir, que casi en la totalidad de las áreas de aplicación, una de las mayores prioridades es la necesidad de realizar procesos con datos de forma muy rápida. Las redes neuronales se adaptan bien a esto debido a su posible implementación paralela. Los resultados presentados en este trabajo validan su uso para esta tarea. Este tipo de algoritmos también se pueden combinar con otros en

reconocimiento de tareas a partir de series temporales de actividad ocular con un mayor grado de parecido.

TRABAJO FUTURO

Como trabajo futuro, se podría combinar el reconocimiento del sonido o voz con el seguimiento de ojos para caracterizar actividad del usuario. Existe un sistema de seguimiento de voz para una sala de videoconferencia corporativa (Anexo F), en donde se dispone de una cámara conectada a una pantalla de ordenador grabando la sala. Ésta funciona con la voz, es decir, graba a aquellas personas que están hablando en ese preciso instante, así por ejemplo, si un sujeto A está diciendo algo, la cámara se centra en grabarle a él, y si es interrumpido por un sujeto B, pasará a grabar al individuo B. De este modo, en la pantalla de grabación sólo saldrán las personas que estén hablando en ese momento. Con esta tecnología explicada y llevándolo al ámbito de la clasificación y detección de actividades, se podría hacer una clasificación de las distintas actividades que lleva a cabo cada sujeto mientras habla delante de la cámara utilizando tanto la información del seguimiento del cuerpo, como la de la mirada y la del reconocimiento del sonido.

La detección de actividad a partir del seguimiento de los ojos tiene aplicaciones en el ámbito biomédico, como por ejemplo el diagnóstico de la dislexia, de los déficits de atención, del estudio de la epilepsia etc. (Boraston & Blakemore, 2007). También tiene aplicaciones en el ámbito de los interfaces cerebro-máquina donde se puede correlacionar la actividad de la mirada con los registros del electroencefalograma (EEG) (Lee et al., 2013; Plöchl, Ossandón, & König, 2012). En este ámbito la caracterización de la actividad de los ojos es también útil para la eliminación de artefactos en el registro de la señal de EEG por el movimiento ocular.

En este trabajo se ha clasificado cuatro tipos de actividades a partir del seguimiento ocular para ilustrar este tipo de aplicación. El estudio se puede expandir a muchos otros tipos de actividades relacionadas con el control parental, el ajuste automático de parámetros de imagen, sonido, etc. relacionados con la optimización de

una aplicación en función del contexto de su uso, y la identificación biométrica de los usuarios.

REFERENCIAS

Boraston, Z., & Blakemore, S.-J. (2007). The application of eye-tracking technology in the study of autism. *The Journal of Physiology*, *581*(Pt 3), 893–898.

Diferentes aplicaciones de la tecnología Eye Tracking. (2004). Retrieved from http://www.ergoestudio.com/articulos/articulos/diferentes_aplicaciones_et.php

Duchowski, A. (2007). *Eye tracking methodology: Theory and practice*. *Eye Tracking Methodology: Theory and Practice* (pp. 1–328). Springer London.

Duchowski, A. T. (2002). A breadth-first survey of eye-tracking applications. *Behavior research methods, instruments, & computers : a journal of the Psychonomic Society, Inc*, *34*(4), 455–470. doi:10.3758/BF03195475

Eyetracking. (2013). Retrieved from http://docudigitalmaster.wordpress.com/usabilidad/temas-unidad/eyetracking/

George, D., & Hawkins, J. (2009). Towards a mathematical theory of cortical micro-circuits. *PLoS Computational Biology*, *5*(10).

Granka, L. A., Joachims, T., & Gay, G. (2004). Eye-Tracking Analysis of User Behavior in WWW Search. *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval* (pp. 478 – 479). doi:10.1145/1008992.1009079

Guerrero, J. (2013). SERVICIO DE ALMACENAMIENTO PARA COMPARTIR Y SINCRONIZAR ARCHIVOS ENTRE ALUMNOS Y DOCENTES DE LA UNIVERSIDAD NACIONAL EXPERIMENTAL DE LOS LLANOS OCCIDENTALES “EZEQUIEL ZAMORA” UNELLEZ – BARINAS. Retrieved from http://es.scribd.com/doc/218171160/TESIS-JULIOGUERRERO-MANUELAZUAJE

Lee, J. H., Lim, J. H., Hwang, H. J., & Im, C. H. (2013). Development of a hybrid mental speller combining EEG-based brain-computer interface and webcam-based eye-tracking. *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS* (pp. 2240–2242).

Olsen, A., & Marshall, P. (2011). Using Eye Tracking for Interaction. *Human Factors*, 741–744. doi:10.1145/1979742.1979541

Plöchl, M., Ossandón, J. P., & König, P. (2012). Combining EEG and eye tracking: identification, characterization, and correction of eye movement artifacts in electroencephalographic data. *Frontiers in Human Neuroscience*.

Poole, A., & Ball, L. J. (2005). Eye Tracking in Human-Computer Interaction and Usability Research: Current Status and Future Prospects. In C. Ghaoui (Ed.), *Encyclopedia of Human-Computer Interaction* (pp. 211–219). Idea Group. doi:10.4018/978-1-59140-562-7

Rozado, D, Moreno, T., San Agustin, J., Rodriguez, F. B., & Varona, P. (2014). Controlling a Smartphone Using Gaze Gestures as the Input Mechanism. *Human–Computer Interaction*, In press. doi:10.1080/07370024.2013.870385

Rozado, David, Agustin, J. S., Rodriguez, F. B., & Varona, P. (2012). Gliding and Saccadic Gaze Gesture Recognition in Real Time. *ACM Transactions on Interactive Intelligent Systems*, *1*(2), 1–27. doi:10.1145/2070719.2070723

Rozado, David, Rodriguez, F. B., & Varona, P. (2012a). Low cost remote gaze gesture recognition in real time. *Applied Soft Computing*, *12*(8), 2072–2084. doi:10.1016/j.asoc.2012.02.023

Rozado, David, Rodriguez, F. B., & Varona, P. (2012b). Extending the bioinspired hierarchical temporal memory paradigm for sign language recognition. *Neurocomputing*, *79*, 75–86. doi:10.1016/j.neucom.2011.10.005

Technology, T. (2013). Tobii X2-30 Eye Tracker Flexibility of eye tracking study designs.

Tobii. (2012). Tobii_X2-30_EyeTrackerUserManual_WEB.

Tribe, T. E. (2014). MWC 2014 : The Eye Tribe o cómo jugar a Fruit Ninja con la mirada.

¿Qué es el “Eye Tracking” y para qué nos sirve? (2012). Retrieved from http://blog.solucionesc2.com/que-es-el-eye-tracking-y-para-que-nos-sirve-c2-usabilidad-web

ANEXO A: Texto a leer

En este anexo se incluye el trozo de texto que se puso a los participantes de las pruebas:

Aunque no se conoce bien el mecanismo, parece que hay dos tipos de personas: las que recuerdan con frecuencia sus sueños y las que apenas los recuerdan. Para añadir más interés al asunto, un estudio acaba de confirmar que el cerebro de unos y otros funciona de manera diferente, y que en estas sutiles diferencias podría estar la clave entre recordar y no recordar lo soñado.

El estudio ha sido realizado por **Perrine Ruby y su equipo** del Centro de Investigación en Neurociencia de Lyon (Francia), quienes realizaron una serie de pruebas a un grupo de 36 sujetos sanos en un laboratorio del sueño. La mitad de ellos eran de los que apenas recuerdan lo soñado (uno o dos sueños al mes como mucho) y el resto pertenecía a esa clase de personas que recuerda sus sueños casi a diario. Todos ellos pasaron una noche en el centro mientras los científicos monitorizaban su actividad cerebral con encefalogramas y emitían una serie de sonidos a los sujetos a través de auriculares para comprobar sus reacciones.

La prueba principal consistía en **llamar por su nombre de pila a cada sujeto** por los auriculares - además de pronunciar el nombre de un desconocido - y comprobar qué pasaba. Lo que vieron fue que, en respuesta a sus nombres, los sujetos con buena memoria para los sueños mostraban unas **ondas cerebrales diferentes**, con mucha más actividad. "Parecían ser mucho más reactivos al ambiente", asegura Perrine Ruby en Popular Science. Es decir, su cerebro reaccionaba de manera activa mientras les llamaban, mientras que en el caso de los que no suelen recordar sus sueños seguían durmiendo como troncos.

ANEXO B: Video 1

El primer video que se mostró a los participantes fue un video tranquilo de una playa donde se puede ver pinchando el siguiente enlace:

https://www.youtube.com/watch?v=le4auqd6VC4

Solo se mostraron los primeros 35 segundos.

ANEXO C: Video 2

El segundo video mostrado a los sujetos fue un video de acción, del minuto 1:22 al minutos 1:57. Puede verse pinchando en el enlace:

https://www.youtube.com/watch?v=bKQYK7PYQpQ

ANEXO D: Navegación

Como ya se mencionó anteriormente, la navegación fue realizada en la página web de la Universidad Autónoma de Madrid, desde el enlace: www.uam.es

Desde ese link, al usuario se le fueron dando indicaciones de cómo tenía que hacer la navegación y qué *links* pinchar.

- ✓ **Paso 1:** Entrar en "Facultades". Esta pestaña aparece arriba en verde.
- ✓ **Paso 2:** Una vez dentro, pinchar en "Facultad Politécnica Superior". Este enlace aparece centralizado, en letras grandes y verdes.
- ✓ **Paso 3:** pulsar "Acceso a su web". También aparece centrado y en verde.
- ✓ **Paso 4:** Entrar en la pestaña de "Estudios". Aparecerá un desplegable y se pulsará en "Grado".
- ✓ **Paso 5:** Una vez dentro, se pinchará en "Grado en Ingeniería Informática".
- ✓ **Paso 6:** Después se entrará a "Horarios".
- ✓ **Paso 7:** En el centro aparecerá una tabla con los horarios. Se deberá pulsar en "Cuarto curso".
- ✓ **Paso 8:** Nuevamente, en el centro, aparecerán asignaturas. Se deberá buscar "Simulación" y *clickar* en ella.

Estos pasos son cronometrados para ser realizados en 30 segundos. Hay gente capaz de realizarlo en ese tiempo y otros no.

ANEXO E: *Audacity*

Aquí se adjunta la página web de Audacity, para poder descargarlo, verlo y trabajar con ello: http://audacity.sourceforge.net/

ANEXO F: Seguimiento de Voz para Sala de Video-Conferencia Corporativa

Este enlace contiene una muestra de lo explicado anteriormente sobre el control de una cámara mediante el seguimiento de voz.
http://www.youtube.com/watch?v=ERTNiEoEx6E